An Architecture for Secure Document Flow & Archival Systems

BY
Hussam Eddin Abdullah Al-Sawadi
hussam@ccse.kfupm.edu.sa

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

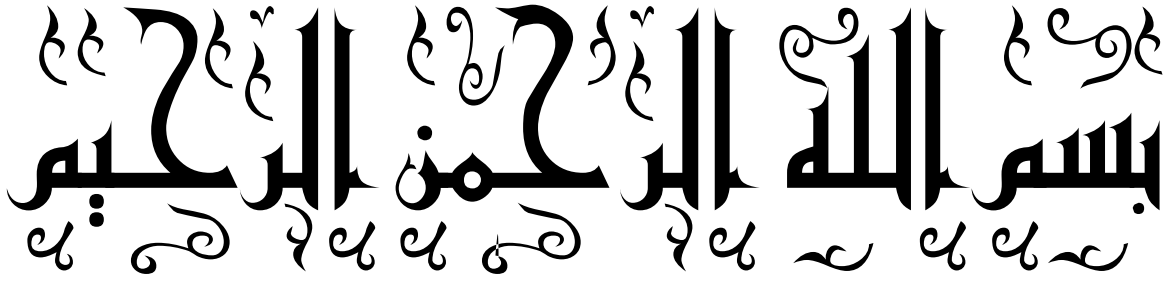
In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE


بسم الله الرحمن الرحيم

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

HUSSAM EDDIN ABDULLAH AL-SAWADI

under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Thesis Committee

_____
Dr. Muhammad S. Al-Mulhem (Chairman)

_____
Dr. Kanaan Faisal (Member)

_____
Dr. Muhammad Shafique (Member)

_____
Department Chairman, Dr. Kanaan Faisal

_____
Dean of Graduate Studies , Dr. Mohammad A. Al-Ohali

_____
Date

ACKNOWLEDGMENTS

First and foremost, all of my thanks and praises are to my god, Allah, for his boons and helps during my life. He prepared the suitable circumstances for my M.Sc. study and research that I did not expect. This research can not appear without his desiring.

Second, since the Prophet once said "Who does not thank people, he does not thank Allah", I would like to express my deepest appreciations to the chairman of the thesis committee, Dr. Muhammad Al-Mulhem, for his effective advice and support. Also, it gives me great pleasure to appreciate the other committee members, Dr. Kanaan Faisal and Dr. Muhammad Shafique, for their fruitful remarks and comments. Moreover, I would like to acknowledge Dr. Subbarao Ghanta for originating the idea of this thesis, reviewing the first draft of the developed model and teaching me four courses during my master level. I also wish to thank Dr. Jarallah Al-Ghamdi for his comments on this work.

Also, my warmest and most sincere thanks go to my family members, especially my mother, father and wife. I am very appreciative for their care, encouragement and sacrifice.

Last but not least, I would like to thank all those who helped me in this research even though I did not mention their names. May Allah bless this work and benefit humankind with it?

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

Name: **HUSSAM EDDIN ABDULLAH ALSAWADI**

Title: **AN ARCHITECTURE FOR SECURE DOCUMENT FLOW & ARCHIVAL SYSTEMS**

Degree: **MASTER OF SCIENCE**

Major Field: **INFORMATION & COMPUTER SCIENCE**

Date of Degree: **JUNE 2005**

Most document collaborative systems enclose implied document flow models which are suitable only for their systems. Moreover, these systems can concentrate on their foremost objectives if they utilizes generic document flow model. In this research, I have investigated the development of a generic architecture for Document Flow model based on WebDAV protocol (DFWDAV), which are intended to overcome the limitations of the specialized models. Moreover, this model can be utilized as an infrastructure for document systems.

Keywords: *Document Flow model based on WebDAV protocol (DFWDAV), Document Flow, Cooperative Systems, Office Automation, World Wide Web Distributed Authoring and Versioning (WebDAV) Protocol.*

King Fahd University of Petroleum and Minerals, Dhahran.

JUNE 2005

THESIS ABSTRACT (ARABIC)

ملخص الرسالة

اسم الطالب الكامل: حسام الدين عبدالله السوادي

عنوان الدراسة: هيكل لنظم تدفق وأرشفة الوثائق

الدرجة: ماجستير علوم

التخصص: معلومات وعلوم الحاسب الآلي

تاريخ الشهادة: جمادى الأولى 1426هـ

معظـــم أنظمة الوثائق التعاونية تحتوي على نماذج تدفق وثائق ملائمة لهذه الأنظمة. وعلاوة على ذلك فهـــذه الأنظمة يمكن أن تركز على أهدافها الأساسية إذا استفادت من نموذج عام لتدفق الوثائق. نقدم في هذه الرســـالة تطويـــر لهـــيكل عام لنموذج تدفق وثائق معتمد على بروتوكول التأليف الموزع والنُسخ عبر الشبكة العنكبوتـــية العالمية بحيث يتلافى عيوب النماذج الخاصة. إيضاً فهذا النموذج يمكن أن يستغل كبنية تحتية لأنظمة الوثائق.

مفاتـــيح: نموذج تدفق الوثائق المعتمد على بروتوكول التأليف الموزع والنُسخ عبر الشبكة العنكبوتية العالمية – تدفـــق الوثـــائق – الأنظمة التعاونية – أتمتة المكاتب – بروتوكول التأليف الموزع والنُسخ عبر الشبكة العنكبوتية العالمية.

جامعة الملك فهد للبترول والمعادن، الظهران.

جمادى الأولى 1426هـ

CHAPTER 1

INTRODUCTION

The document flow across the Web has become an essential task in document collaborative systems which serve distributed offices and working groups. Moreover, documents are used as a means of information interchange and they are designed to explicitly serve the function of effective communication. On the other hand, computerized document collaborative systems are emerging technologies. Furthermore, several such systems use built-in document flow functions. Some examples of these applications are cooperative authoring, report evaluating and proposal requesting.

1.1 BACKGROUND

The selected subject deals with several issues and makes use of results from several different areas. These include document management systems, XML language and WebDAV protocol. This section includes brief descriptions of these technologies as regards the background.

At the outset, several definitions of a document are considered. After that, three main models for documents are stated. Then, brief discussions about

document management systems and document flow types are presented. Next, brief information about XML is offered. This information includes a short history of markup languages, provides a brief overview of XML, shows how a structured document can be described by XML by providing different templates for several document types, and defines some terminology related to XML. Finally, an introduction to the contemporary communication protocol WebDAV is provided.

1.1.1 Document Procedural Definition

There are many definitions of a document. The following are some views of document definitions adapted from (1):

- **It is any container of coherent information which has been assembled for human understanding.**
- **It is a notation that we have invented for storing information, so that we do not forget it.**

Moreover, a document is defined as (2):

- **It is a set of information pertaining to a topic, structured for human comprehension, represented by a variety of symbols, stored and handled as a unit.**

A container and storage of information are very general phrases since they vary in their formats (e.g. papers, videotapes, computer files, etc.). Yet, for the purpose of this study, I will focus on computerized documents. From the low-level view of computers, those documents are collections of binary digits. On the other hand, the

outside appearances of computerized documents can take several diverse shapes, depending on their high level purposes. The following list includes different types of possible computerized documents (3), (4), (5), (6), (7), (8), (9):

- **Articles.**
- **Source code.**
- **Request for proposals.**
- **Proposals.**
- **Evaluation reports.**
- **Mathematical equations.**
- **Scientific papers.**
- **Books.**
- **Contracts and agreements.**
- **Drawings, blueprints and photographs.**
- **Reports.**
- **Email and voicemail messages.**
- **Manuals and handbooks.**
- **Business forms.**
- **Presentations.**
- **Correspondence.**
- **Memos.**
- **Transcripts.**

In spite of the fact that document types affect most document systems, it is possible to develop a document flow model based on WebDAV protocol which will not depend upon document types.

1.1.2 Computerized Document Models

There are several models of computerized documents. Three main models have been used according to (10):

1- **Unstructured sequence of text: The documents in this model are represented as lengthy sequences of text.**
2- **Semi-structured data with nodes and links: The documents in this model are treated as graphs with nodes and links with strange structure.**
3- **Fully structured data with known structure schema: The documents in this model are represented by a well-defined structure.**

1.1.3 Document Management Systems (DMS)

The document management is defined as "the process of managing documents through their lifecycle from inception through creation, review, storage and dissemination all the way to their destruction." (1).

Document management systems encompass several key components, each of which can be thought of as a stand-alone concept, but together they contribute to a

powerful inclusive system for the management of information. Document management functions include (1), (10), (11):

- **Storage and retrieval.**
- **Status reporting.**
- **Access control.**
- **Indexing.**
- **Version control.**
- **Workflow management.**
- **Document conversion.**
- **Commenting.**
- **Link management.**
- **Linguistic analysis.**

Moreover, since the Web attracts most application areas, document management systems are turning to the Web environment (6), (12).

1.1.4 Document Flow Types

There are several types of document flows such as sequential, parallel and branching flows as shown in Figure 1.1 . The simplest type of them is the sequential flow in which documents route through linear steps. The second flow type is the parallel flow where documents can route to several users at the same time as illustrated in the same figure. Finally, the branching flow is a conditional path flow

based on some conditions which can be related to time, user operations or system actions.

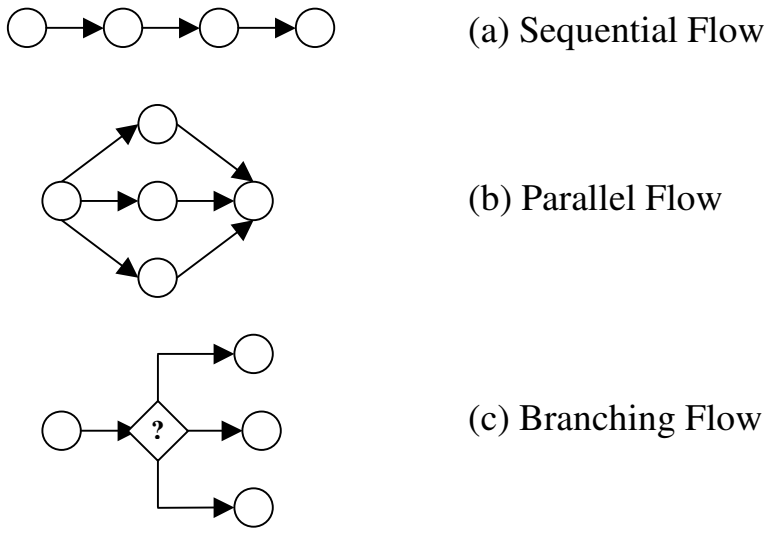


Figure 1.1: Document Flow Types

1.1.5 Short History Of Markup Languages

Markup languages are used to describe documents (13). Although there are several markup languages, it is expected that eXtensible Markup Language (XML) will dominate the scene in the coming years. In the 1960s, IBM created a markup language that was called General Markup Language (GML). The purpose of that language was to solve the problems resulted from interchanging documents between different platforms. In the 1980s, the International Organization for Standardization (ISO) used GML to produce the Standard Generalized Markup

Language (SGML) as an official standard (ISO 8879). SGML is a notation for representing documents and making their inherent structure explicit. Many large organizations have used SGML for describing documents. SGML is a robust and strong language; despite this fact it is complex and expensive to implement and maintain and hence it is not suitable to be readily used.

Several languages are derived from SGML such as the ubiquitous Hypertext Markup Language (HTML) and its generalizing descendent XML. HTML was created in the late 1980s. Its purpose was to unify the way of storing and retrieving data between researchers. It has been built on the concept of hypertext in which one can link documents and transfer between them by clicking parts of text. In the early 1990s, HTML became a major player in the information exchange among people all over the world because most documents on the World Wide Web (WWW) were described by HTML. Since HTML was not originally designed to be used by WWW, several limitations of HTML became noticeable limitations. The obvious suitable alternative would be XML language (13), (14).

1.1.6 XML

The eXtensible Markup Language is a meta-language defined by the World Wide Web Consortium (W^3C), derived from SGML, that can be used to describe a broad range of hierarchical markup languages. It is described and defined as a W^3C Recommendation (15). XML has a standard syntax that is flexible enough to represent just about any kind of information (14). The rich tools of XML can be

operated together to provide modern software architectures (e.g. architecture for a document management system) (12).

XML Features:

- **It is an extensible syntax; hence it is not limiting anybody.**
- **XML is similar enough to HTML which enables one to learn its basics in a short time.**
- **XML's popularity and support from the industry means there is no shortage of expertise and products to support its development.**

XML Document Syntax:

An XML "document" consists of a *prolog*, *processing instruction*(*s*), a *root element* (with optional *attributes*) and a hierarchy of *sub-elements* (optionally with *attributes*), *entities*, and (parsed) *character data*. XML has no predefined elements, and it is left to the author to define those elements that make sense for his application (14).

An XML Document starts with a *prolog*. The minimal prolog contains a declaration that identifies the document as an XML document. The declaration may also contain additional information. The prolog can also contain definitions of entities and specifications. An entity is an item that is inserted when one references it from within the document. A specification tells which tags are valid

in the document. Both the entities and the specifications are declared in a Document Type Definition (DTD) that can be defined directly within the prolog, as well as with pointers to external specification files.

An XML Document can also contain *processing instructions* that give commands or information to an application that is processing the XML data. Processing instructions have the following format: `<?target instructions?>` **where the** `target` **is the name of the application that is expected to be doing the processing, and the** `instructions` **are a string of characters that embodies the information or commands for the application to be processed.**

Each XML document has a *root element* that consists of *elements* delimited by *tags*. *Attributes* can be attached to elements, as part of their opening tag. Elements can be nested and thus form a nice hierarchical representation.

XML uses the HTML angular brackets (i.e. < and >) as tag delimiters but XML tags specify what the data means, rather than how to display it. Also, XML has an accurate syntax: every opening tag (e.g. <someTag>) must have its corresponding closing tag (e.g. </someTag >). There should not be any cross (i.e. which tag is opened first, should be closed first) between tags. Finally, XML tags are case-sensitive and so <tag> is different from <TAG>. On the other hand, empty tags use special delimiters (e.g. <someEmptyTag/>) that combine the closing and opening tags.

XML supports two different types of documents: well-formed and valid. Well-formed documents simply respect the syntax defined by the XML

specification. Valid documents not only respect the syntax but also define a limited set of tags in a DTD. The DTD defines which elements are authorized and where. For example, the DTD may prescribe that `<alternate>` **elements can appear in** `<email>` **elements but not in** `<name>` **elements.**

Example:

Here is an example of some XML data which might be used in a messaging application:

```
<message>
    <to>you@yourAddress.com</to>
    <from>me@myAddress.com</from>
    <subject>XML Is Really Nice</subject>
    <text>
       In how many ways is XML nice?
        Let me count the ways...
    </text>
</message>
```

Applications of XML:

XML is being used for a broad variety of applications including:

- **Vertical markup languages for mathematics, chemistry, and other document-centric publishing applications.**
- **E-Commerce solutions.**

- **Intra-, extra-, and inter-enterprise application messaging.**

Indeed the most popular applications for XML are not just as fancy replacements of HTML but also as a general format for data-exchange. It is expected that XML will become the standard for data interchange on the Web (16).

1.1.7 Using XML for Structuring Documents

XML, like its parent SGML, is widely acceptable to be used for structured documents (17). It is expected that document applications will take the advantages of XML technologies (6), (16). Several recent applications utilize XML for structuring the documents (3), (4), (18). Moreover, there are several individual experimental works to define XML document type definitions (DTD) for different document types like article, book, proposal, paper, etc (12). At the same time, the direction of standardizing several DTDs is growing fast (14).

1.1.8 XML Document Templates

While XML represents data using tree-structures, XML can represent other data structures because of its adaptability. Several templates for different data structures are presented next (14):

Forms:

Using XML, a form can be represented using the following template:

```
<FormName>
     < field1>
          data1
     < /field1>
     < field2>
          data2
     < /field2>
     .
     .
     .
     < fieldn>
          datan
     < /fieldn>
</FormName>
```

Tables:

Using XML, a table can be represented using the following template:

```
<TableName>
<row1>
     <col1>
          data[1,1]
     </col1>
     <col2>
          data[1,2]
     </col2>
     .
     .
     <coln>
          data[1,n]
```

```
     </coln>
</row1>
<row2>
     <col1>
          data[2,1]
     </col1>
     <col2>
          data[2,2]
     </col2>
     .
     .
     <coln>
          data[2,n]
     </coln>
</row2>
.
.
.
<rowm>
     <col1>
          data[m,1]
     </col1>
     <col2>
          data[m,2]
     </col2>
     .
     .
     <coln>
          data[m,n]
     </coln>
</rowm>
```

```
</TableName>
```

1.1.9 XML Related Terminology

This subsection identifies several terminologies which are related to XML, such as DTD, XSL, CSS, SAX and DOM.

DTD:

A Document Type Definition (DTD) provides a specification for an XML document. It specifies structural elements and markup definitions that can be used to create documents (14), (16).

XSL & CSS:

As stated above, one of the best features of XML is the separation of the content from the display. Therefore, formatting attributes are not part of XML documents or DTDs. Instead, XML can rely on either eXtensible Style Language (XSL) or Cascading Style Sheets (CSS). Both XSL and CSS are style-sheet mechanisms, which provide browsers with formatting and displaying information. XSL is an advanced XML-specific style-sheet mechanism whereas CSS applies the HTML style-sheet mechanism (13), (19), (20), (21).

SAX & DOM:

The most popular APIs (Application Program Interfaces) which are used to process XML documents are SAX and DOM.

The Simple API for XML (SAX) is an event-driven API. It generates events such as the start or the end of an element. Depending on its tasks, each application uses the appropriate events.

The Documents Object Model (DOM) converts an XML document to a hierarchical tree. The nodes of the tree represent the document's elements and content. The tree can be easily accessed and processed by an application or programming language.

Since DOM provides the hierarchical trees of XML documents, it requires more memory than SAX which does not need any data structure. On the other hand, while DOM is more flexible because it is suitable for applications which use an XML document as a whole, SAX is more efficient because it can be used with giant documents which do not fit in memory (16), (22), (23).

1.1.10 WebDAV

The developed document flow model, namely Document Flow model based on WebDAV protocol (DFWDAV), is completely established over the World Wide

Web Distributed Authoring and Versioning (WebDAV) protocol. This novel protocol is under current development by the WebDAV working groups in the Internet Engineering Task Force (IETF) and it is an extension to the Hypertext Transfer Protocol (HTTP). The WebDAV extension defines a standard infrastructure for collaboration across the Web. WebDAV makes the Web as a writeable, collaborative medium. Today, collaborating people in geographically distant locations interchange information through documents. Usually, they use emails to notify each other of changes to documents, fill directories with revisions and mail questions about what is finished and what is still under construction. It is obvious that this method of collaborative work necessitates superfluous efforts and consumes time. In contrast, WebDAV provides several features which simplify the teamwork on the Web. Those features include supplying document properties like author and creation date, editing Web documents in place, providing a locking mechanism to prevent the lost update problem, applying access privileges and supporting versioning (9), (24), (25).

Using WebDAV as an infrastructure for a document flow system provides several advantageous features such as simplifying works, reducing efforts and removing the bulk of responsibility from the applications (8), (26).

Comprehensive but recapitulated information on WebDAV protocol is presented in detail in Chapter 4 as extra information is required to empathize the DFWDAV model.

1.2 PROBLEM DESCRIPTION

Most contemporary organizations use manual document flow systems. First, an employee initiates a document. Then, such a document flows (routes) through a specific path of other employees. These employees browse, revise, update, modify or finalize the document. The result is several different versions originated from one document. The problems of out-dated, incomplete, inaccurate, hard to maintain, lost documents etc. are all attributed to manual document flow systems. Consequently, it is difficult to keep track of changes or to compare different versions of a document. Even the task of determining the current status of a document (current location, authors, changing dates, etc.) can be a demanding task, especially in large organizations. In summary, several users tend to collaborate to produce documents (8), (9), (26).

Several problems may be considered as special cases of the previous problem, depending on the organizations, their users and the nature of the documents. For example, the conference management problem (6) is basically a special instance of the described problem. The organization in the conference management problem is a scientific community that manages conferences. The conference management problem documents are the papers which are submitted to the conferences. The documents in a conference flow between the authors, the referees, the conference editor and the chair.

Another special case of the thesis problem is the production of documents in the context of an electronic market application, namely a platform dealing with

requests for proposals (3). In this problem, several users at the buyer side and the supplier side collaborate to produce several kinds of documents. Those documents include proposals, requests for proposals, evaluation reports and business descriptions.

The final example of the document flow is the collaborative authoring systems (8), (27), (28), (29). The users of the collaborative authoring systems are the authors, reviewers and editors. These users cooperate to create, check, improve and confirm authoring documents such as books, scientific papers or even poems.

1.3 AIMS AND OBJECTIVES OF THIS THESIS

Within the scope of the thesis work, the researcher aims to design and provide a high-quality document flow model based on WebDAV protocol. This model provides a document flow infrastructure for different document collaborative systems which support multi-user works. The major functions of the model include:

- **Creating flow definitions.**
- **Assigning flow definitions to documents.**
- **Automating document flows between users according to the assigned flow definitions.**
- **Analyzing document flows.**
- **Notifying users by appropriate information via emails.**
- **Providing historical information about the actual flows of documents.**

1.4 ORGANIZATION OF THE REMAINING CHAPTERS

The remaining chapters of this thesis are arranged as follows. Chapter two is a brief presentation of several document flow models in different applications. The DFWDAV architecture is shown in Chapter 3. Since the DFWDAV architecture is established over WebDAV protocol, Chapter 4 presents a short and concise study of its workgroups, functions, and status. Chapter 5 focuses on the implementation issues. In Chapter 6, a comprehensive comparison between the studied and DFWDAV models is presented. Finally, Chapter 7 closes the thesis by the conclusions and recommendations.

CHAPTER 2

LITERATURE SURVEY

In this chapter, I will present a summary of several document flow and archival models in different groupware systems which have a strong relation to the present study. Most of these groupware systems provide extensive functionality because they were designed by great and impressive efforts extended for several years by dedicated teams. However, for the purpose of this study I will concentrate on the research subject of document flow and archival models. Fifteen different models are discussed in a chronological order because time is an important factor in the progress of this area. This will be settled in the Analysis and Comparison chapter.

2.1 ENGINEERING DOCUMENT MANAGEMENT SYSTEM (EDMS)

This system was developed as a joint project between the department of computer science in Helsinki University of Technology in cooperation with KONE Elevators which is a large Finnish elevator manufacturer (30), (31). The objective of this system is to manage engineering documents such as engineering drawings. The system architecture is based upon the client/server architecture, the Unix operating

system and the LAN network. The server manages a relational database that contains the documents and their attributes. The clients require special software (tool), which is used to read and write the documents from and to the server. A user who wants to read or to modify a document must first lock it on the server and then copy it to an ordinary local file using the software tool. When the user finishes the modification, he copies the file back to the server and removes the lock. Figure 2.1 shows the system plan. It is possible to restrict the user access (read and write) to the documents by defining the access rights on the documents for each user.

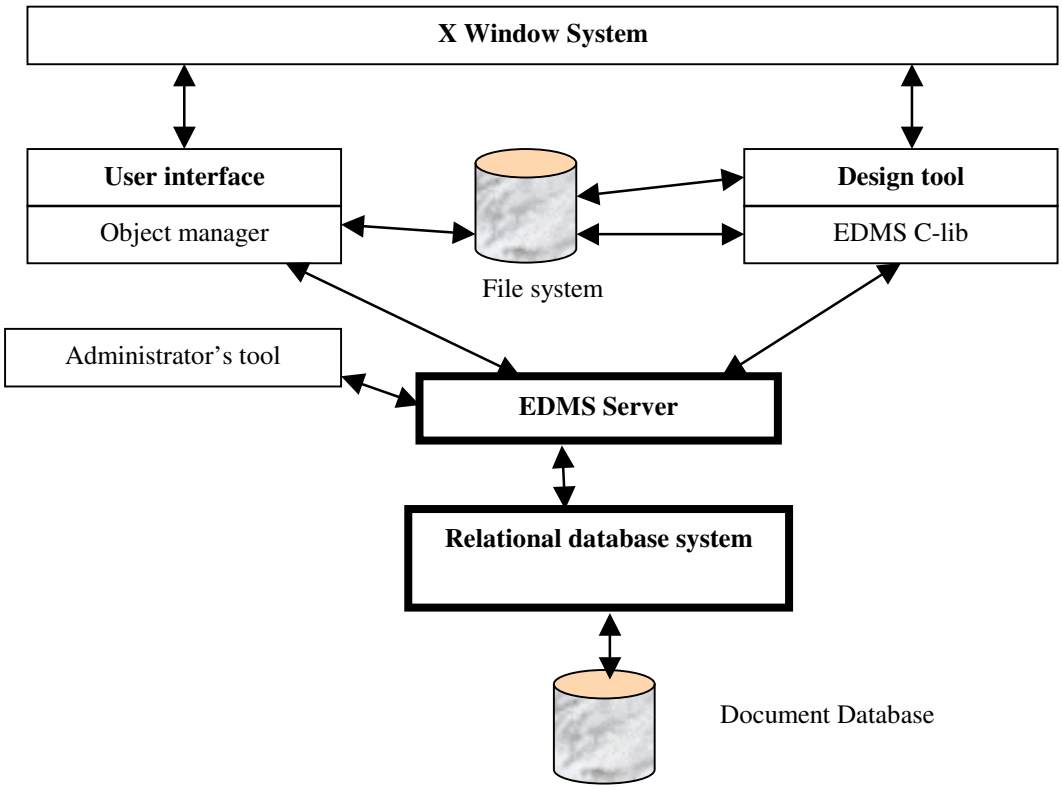


Figure 2.1: EDMS Architecture

The system provides several excellent features. For example, it is possible to define a state graph for the document flows. When a new document is created, it is moved to the first state. An example of the state graph is shown in Figure 2.2. Also,

this system can generate document versions explicitly by the users. A newly created document version is the same as the original version and can be modified until it is used as the parent of another version. Moreover, it is possible to send asynchronous notifications from the server to the clients to inform them about recent changes.

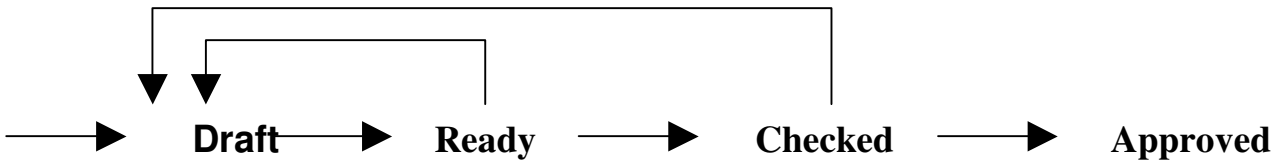


Figure 2.2: An Example of EDMS Document States

2.2 ALLIANCE

Alliance is a structured cooperative application which works on a WAN (Wide Area Network). It allows authors on different Web sites to create and produce documents in a structured manner (32), (33). For each document, users are assigned several roles. These roles are the manager, the writer, the reader and the null roles. They can be assigned to a full document or part of it. The manager role is the highest role because any user with this role can assign and change the user roles. On the other hand, while a user with the writer role can update the document or part of it, a user with reader role can only read it. Finally, the user with null role on a document part does not have any access to it.

The assignment of the document roles to the users can be on any part of the document. So, any user can have different roles on different parts of a document. Each document is divided automatically to several fragments depending on the

distribution of the user roles. Although any user may be given a particular role on a certain document, this user will not be able to play it at any time. The reason behind this drawback is that another user with the same or higher role used it in advance. In other words, only one user at a time can play the writer or manager roles. A document is represented by a set of files which contains document fragments, user roles for each fragment, the order of the document fragments and the current state for each fragment. Each fragment is saved in a separate file. Alliance works using HTTP over client/server architecture with special scripts as shown in Figure 2.3.

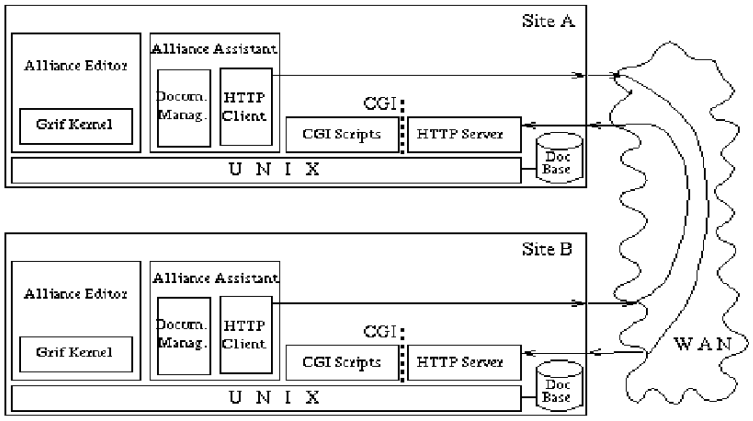


Figure 2.3: Alliance Architecture

A document passes via several phases before it reaches its final shape. First, the document owner creates the list of the users who will work on the document. Second, he sends them a notification message with the necessary information to access the document. Thus, the users can reach the document and contact each other. The document and all needed information are copied on each site where they are required. The needed information includes the document fragments and

management information. Figure 2.4 shows an example of the distribution of the fragment copies of a document. To keep the consistency of the copies, updating a copy should be reflected on the other copies but the updating process of the copies is not necessary to be in the exact time.

Editing a document requires a special treatment to ensure that it is not possible to update the same fragment simultaneously in two different sites. This

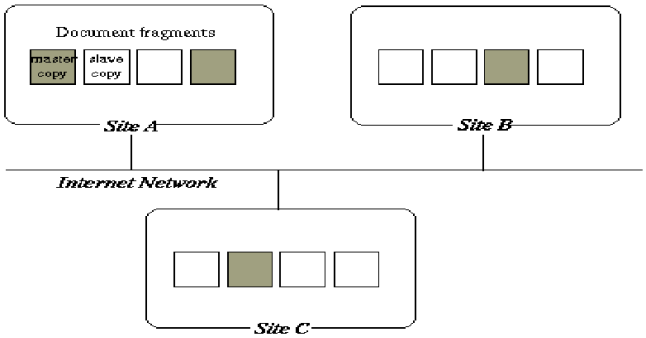


Figure 2.4: Alliance Document Fragments

treatment is based on the notation of master and slave fragments. While there is only one master copy among each set of fragment copies, the rest are slave copies. A master copy is a fragment which a user with a manager or a writer role is working on. Only one user can act with the manager or the writer role on a master copy while several users with the reader or the null roles can work on slave copies. The set of all master copies of the document fragments represents the current state of the document. Therefore, it is obvious that master copies can migrate from one site to another.

2.3 DocMan

DocMan is a document management system which supports distributed cooperative work based on the Internet as shown in Figure 2.5 (11). Each document has an associated document folder (a document store) on a client. Document folders are managed by the distribution and replication service. Each Document folder can hold a set of the document revisions. A document revision is produced when a user updates a document. Updating the document leads the system to mark the original

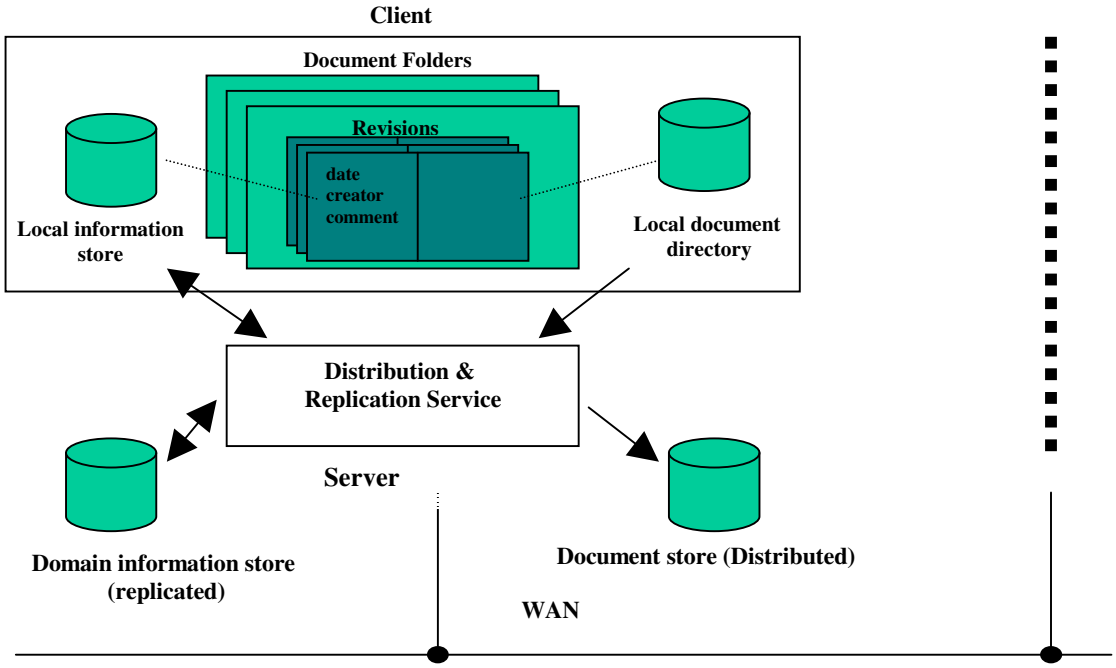


Figure 2.5: DocMan Architecture

document as a new revision and to consider the updated document as the active release. Revisions are always read-only and each document may be revised several times.

The document passes through several stages before it reaches its final shape. Initially, the document folder contains the first revision of the document and the access information. If a user has access to the document, he can fetch it or one of its revisions from the document store to his local disk. The local disk copy is an editable draft copy of the original fetched document. This copy is called a document draft because it is a private working copy of the document. Creating the document draft originates a lock on the document to prevent concurrent modifications. Releasing the draft copy creates a new revision in the local document store and releases the lock. Users can use emails and phones to coordinate their order in fetching the document and making their modifications.

The distribution and replication service is distributed over the site servers. It stores replicates and distributes information about the document folders, the document revisions, the user accesses and the drafts under preparation. Moreover, it stores the revisions in the distributed document store and generates URLs for them. So it is possible to provide public access to the revisions through Web browsers.

2.4 DReSS

DReSS (Document Repository Service Station) is a small software system which allows any Web server to support cooperative publishing (34), (35). Despite the fact that it is designed to be just a document repository by applying the client/server model, it consists of Java programs on both the server and the client sides. However, most of the functionality of the system is on the server side (i.e. the

clients are thin). The client software is any Web browser with two special software helpers to do the following tasks:

- **Start the local suitable editor after downloading the file.**
- **Uploading the document to the server after the user finishes the editing of the document.**

Figure 2.6 shows the system architecture that is composed of the next components:

- **At the client side: an editor, a Web browser and the client-helpers for the purpose of managing the communication between the editor and the browser.**
- **At the server side: a Web server, a repository (database) and the server-helper.**

Figuer 2.6: DReSS Architecture

At the server side, the repository maintains the documents, their meta-data and histories. The documents are represented by ordinary files which exist in the file system of the server. The meta-data contains creator, current author, lock status, authorized authors and log file. Moreover, the server helper handles the communication between the Web server and the repository.

Updating a document requires several steps. These steps include: locking the document, downloading it from the server to the client, editing it in the local machine, uploading it back to the server and finally unlocking it.

One of the DReSS advantages is that it is possible to view the documents from any Web browser when they are public. On the other side, DReSS does not have any versioning capabilities.

2.5 OXFORD RADCLIFFE HOSPITAL DOCUMENT MANAGEMENT

This system was implemented to exchange documents between interested users in the Oxford Radcliffe Hospital NHS Trust (7). The system utilizes two databases on a Web server where the databases represent the document repository. They are the document database which holds the SGML (Standard Generalized Markup Language) documents and the administration database which contains the definition of the users, groups and subscription lists.

The document flow in this system is shown in Figure 2.7. After a system administrator defines the users, groups and subscription lists and he sets up the topics and subtopics, an author can post a document and define its reader distribution list. A reader will receive a notification about a new document, either because he is a member on an author distribution list or because he subscribes in a topic subscription. There is no way to change the documents since the system is not designed to update documents. The main objective of the system is to notify interested users about new information, and so the system does not have any access security. Therefore, all users can view all documents.

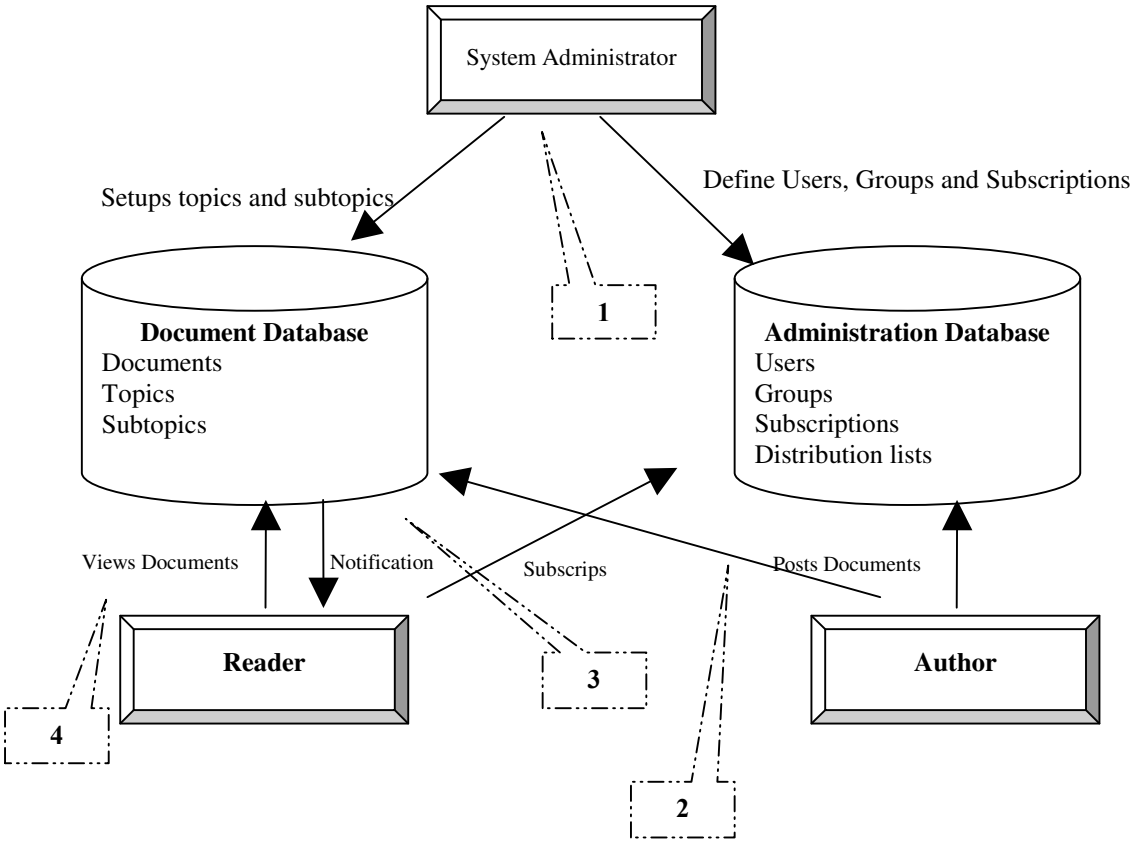


Figure 2.7: The Document Flow in Oxford Radcliffe Hospital Document Management System

2.6 BSCW

The BSCW (Basic Support for Cooperative Work) system has been developed at the German National Research Center for Information Technology (29), (36), (37), (38). This system supports collaboration over the Internet, stores documents, retrieves them and provides users with awareness of the others' activities. The system model is based on the client/server model. The server is an

extension of a Web server to provide the required functionality. In contrast, the client is a Web browser without any modifications. The server provides widely used services such as: document uploading, version management, group administration, access control and document locking. Each group of users establishes a workspace on the server for coordinating their work. The workspace is a repository for the shared information. A user can be a member of several workspaces.

The document repository in this system combines a database and a file system. While the information about the workspaces is stored in the database, the documents are kept as files on the file system of the server.

A user can work on a BSCW system by connecting to BSCW sever using a Web browser. The server generates a suitable HTML page according to the user inputs and selections. At the beginning, he should connect to the server by entering his login name and password. Then, the server returns an HTML page containing all workspaces of which the user is a member. When the user selects a workspace, the server returns an appropriate HTML page showing the content of the workspace.

The document flow in this system is very simple. An author creates a document and uploads it to a workspace. Another privileged user can download the document and edit it locally. The document will be locked after the downloading. After the user modifies the document, he uploads it to the server and unlocks it.

Finally the system provides several interesting facilities, such as version management and event awareness. But the version management functions are too simple. They only provide a linear version scheme on the user requests. The event

service provides useful customizable information about other users' actions. The actions include: reading, uploading and creating new versions of the documents. The event information is generated to each user when he logons to one of his workspaces.

2.7 DOCUMENT FLOW USING SMARTCARD SYSTEM

The objective of this system is to exchange official documents between different government agencies in Taiwan (18). It uses the Internet as a means of communication. In this system, the flow of the documents is too simple because it is only sending the documents from one party to another. But the strong feature of this system is the public-key encryption (39) of documents. It converts an open environment (i.e. the Internet) to a secure means of communication. The system uses XML documents since they can be easily exchanged between different platforms even though it can work with any document.

Figure 2.8 summarizes the document flow in this system. At the sender side, the sender digital signature will be generated using his Smartcard, which will be read by a Smartcard reader connected to his computer. The digital signature requires the private key that is included in the Smartcard. The document which will be sent and the digital signature will be encrypted together by the symmetric encryption using a session key. The session key is randomly generated by a software application. Finally, only the session key will be encrypted using the public key of the receiver. The public key encryption will not take much time since the session key

is a short stream of bits. These operations ensure that nobody except the receiver can decrypt the session key. The combination of the encrypted document and the encrypted session key implements a digital envelope which can be sent to the receiver using the simple mail transfer protocol (SMTP). At the receiver side, the reverse of the sender side operations will be accomplished.

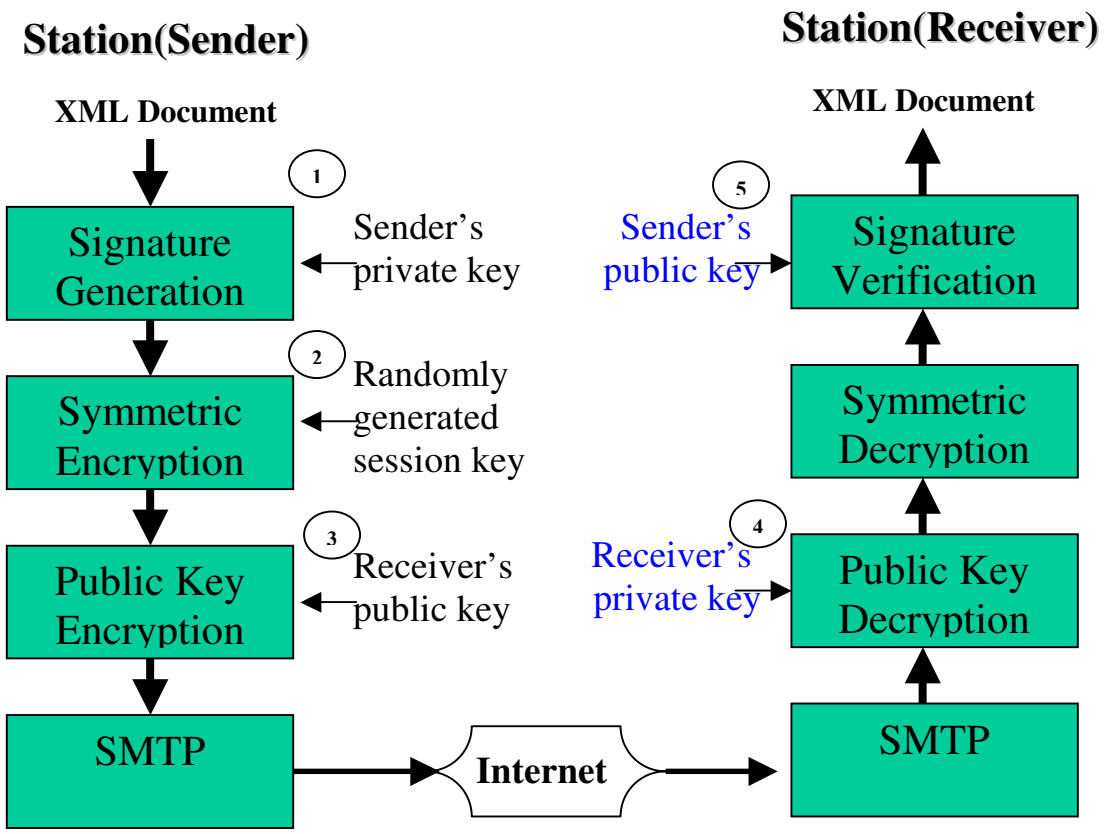


Figure 2.8: Document Flow using Smartcard System

2.8 EnAct

EnAct is an application for drafting, producing and archiving legislation documents which was designed for Tasmanian Department of the Premier and

Cabinet in Australia (40), (41), (42). It utilizes client/server paradigm and an SGML repository (namely Structured Information Manager (SIM)). The repository is resident on the server. It holds the legislation documents as well as the workflow definitions (i.e. process definitions).

The clients are PCs containing special software for communicating, browsing, editing and searching the repository. While this software is a customized version of MS Word, it checks and generates SGML documents.

The system needs special configuration to define the structure of each type of SGML documents. That is to say, the document type definition (DTD) and its elements should be defined for each SGML document type. Also, since the syntax of EnAct documents is SGML format, they can be converted dynamically to HTML format. Therefore, using a Web environment, it is possible to view the documents created by the system using any Web browser.

EnAct provides document versioning capabilities. But it supports only linear versioning as it was designed for legislation documents and only one version of a legislation document should be valid at any time.

Finally, EnAct incorporates with workflow software to provide workflow functionality. For example, once an administrator provides the privileges of updating a document to a user, the user finds the document in his document list.

2.9 DOCUMENT MANAGEMENT WITH INTRANET SETTINGS (AN EXTENSION OF LOTUS NOTES)

This system is a Web based document management system which is an extension of Lotus Notes (43). The objective of this system is documents' handling from the creation up to the production and archiving. Therefore, it defines several stages for the lifecycle of producing documents as shown in Figure 2.9. These stages include the following:

1- **Creating the document.**
2- **Reviewing and approving the document.**
3- **Managing and archiving the document.**

Most of the work of this system depends on Lotus Notes functions, except for the second stage which consists of three phases. These phases are two review cycles and one approval phase. The administrator of the system is responsible for defining which phase should be considered and who is trustworthy about each phase.

The versions are kept in the system unless they are explicitly deleted. Also, the versioning is used to solve the problem of concurrent document editing by different users. Therefore, any user can edit any document at any time. Later on, when he saves the modified document, it becomes a new version of the edited document.

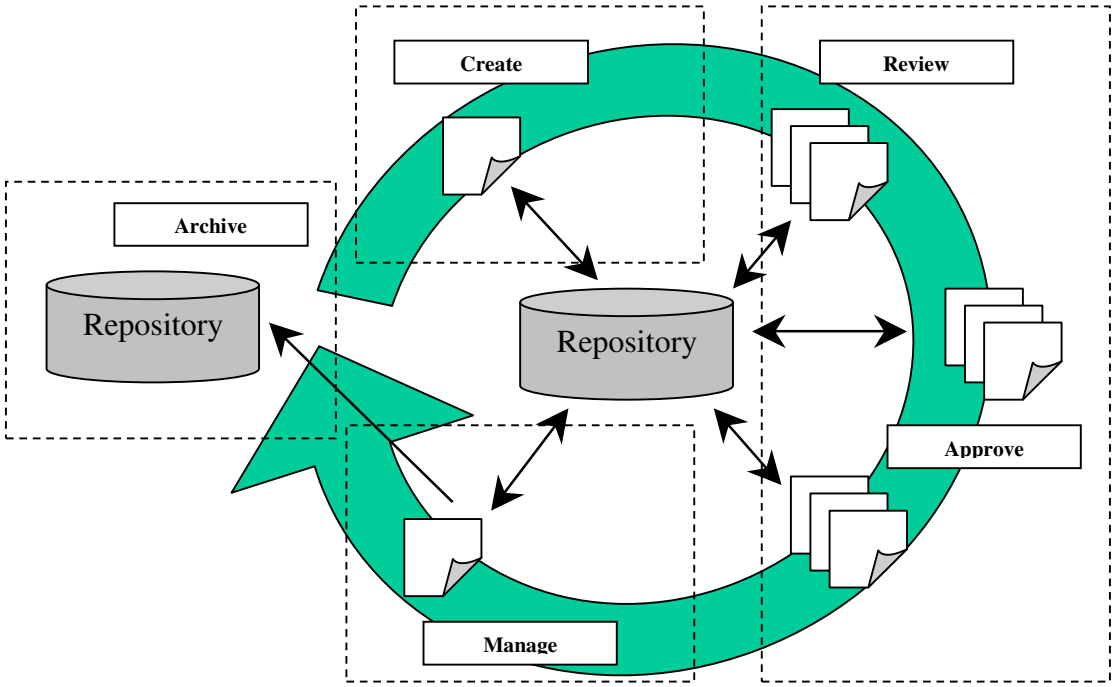


Figure 2.9: The Lifecycle of Producing Documents in the System Extended Lotus Notes

The system interacts with two repositories. While the first repository is used during the creation and the modification of the documents (i.e. in progress documents), the second one archives the finished documents.

2.10 AllianceWeb

AllianceWeb is an extension and modification of Alliance system which is previously explained in section 2.2. It is based on the following principles (28):

-Splitting each document into several fragments.

-Assigning the user roles (manager, writer, reader and null) to the users for each fragment of a document.

-Managing of group awareness.

As Alliance, AllianceWeb is a client/server system. But AllianceWeb is designed for the Web environment. It requires a special extended Web server and a special editor. The editor is an extension to Amaya, which is a public Web-based editor from W^{3}C (World Wide Web Consortium) (44). Also, the client can be any Web browser to view the documents since they are HTML and XML documents.

Figure 2.10 shows the system components of AllianceWeb. AllianceWeb combines both distributed architecture (e.g. Alliance) and cooperative editing model

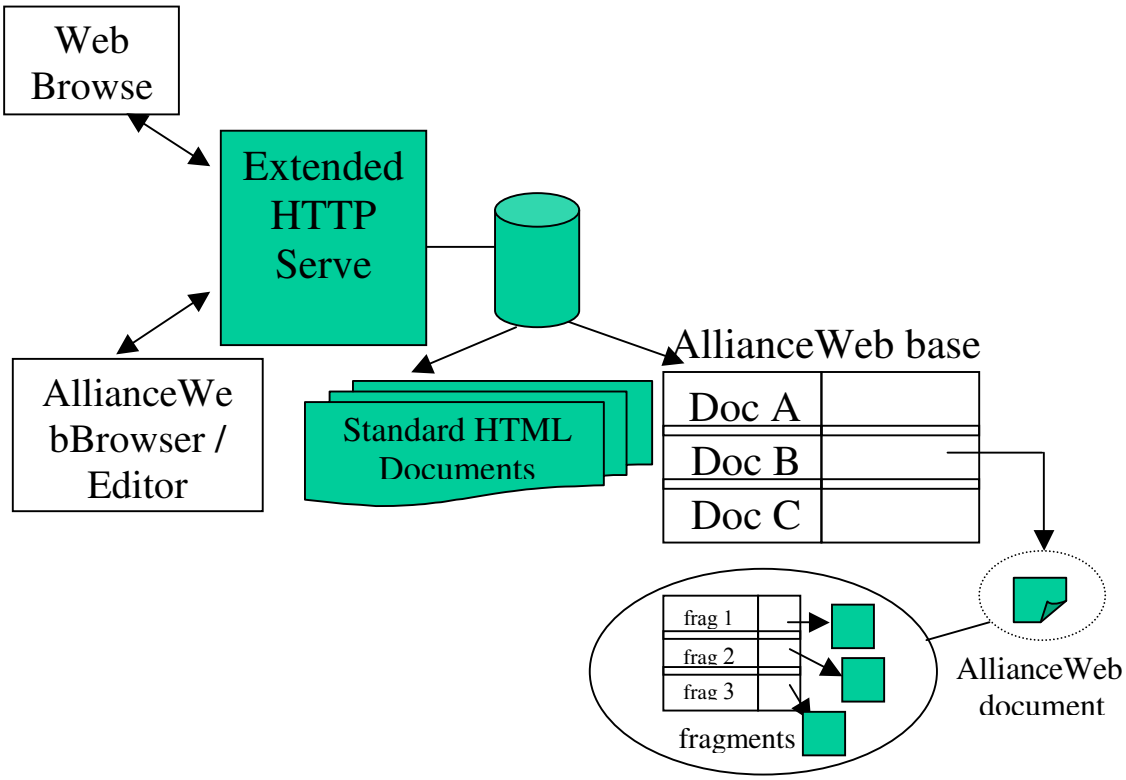


Figure 2.10: The Components of AllianceWeb System

using WebDAV. If there are sites which share a document, and they are connected with unreliable network (i.e. the Internet), the shared document will be replicated

on each site. In contrast, the sites connected with a reliable network (e.g. LAN) share one copy of a document.

2.11 GroupWriter

GroupWriter is an application which supports the collaborative editing of documents (27). It is based upon dividing a document to several sections. An authorized user can create new sections and merge several sections. After outlining a document, the development of the document passes through the following stages as shown in Figure 2.11:

- **Defining the document sections and their owners (sub-team or person).**
- **After the discussion, composing sections by their owners and producing the first draft version.**
- **Reviewing the content of the sections by the team or the owners and providing the suggestions in the form of annotations and revisions.**
- **Making verbal walkthrough by the team and accepting, rejecting or merging the suggestions by the document owners to finalize the document.**

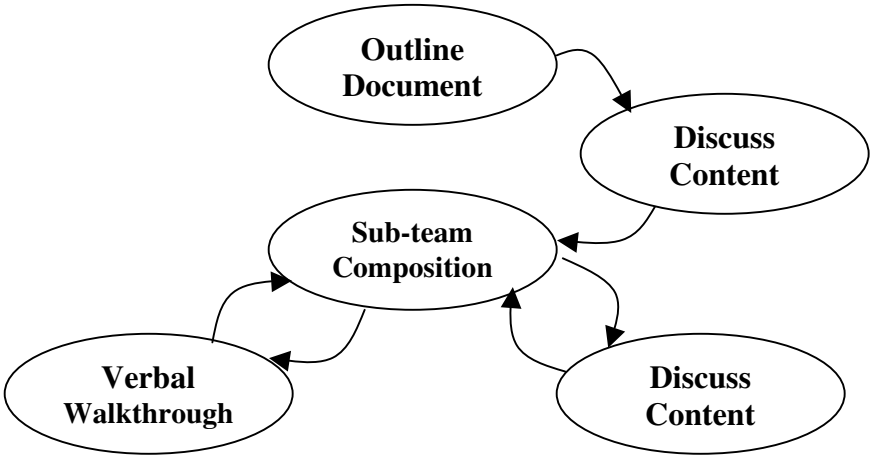


Figure 2.11: GroupWriter Document Development Stages

There are two types of users: local users and dial-up (distributed) users. While the distributed users have limited functionality, the local users can utilize the complete operations of the system. The system allows the users to read any section any time. In contrast, users cannot edit a section simultaneously since it will be locked when a user works with it. Using the table of contents at the edited document, it is possible to know the current writers of all sections. The changes on the sections can be committed on a time base or with checking in, but they will be saved as revisions. If a user connects to the system by the dial-up, he can only provide annotations on the sections. Before finalizing the document, the owners can accept or reject the changes and review the annotations. The system provides several other functions such as: word processing for editing the documents and built-in messaging for communicating with others. It saves documents either in RTF or TXT formats. Because there are plenty of commercial Delphi word processing components, the current version of the system is implemented by Delphi. But there is another version under development by Java to gain Java advantages like portability.

2.12 Web-BASED GROUPWARE SYSTEM BASED on WebDAV PROTOCOL

The infrastructure of this system is based on WebDAV (Web Distributed Authoring and Versioning) (26). WebDAV extends HTTP 1.1 protocol by adding new methods to support collaborative authoring. The system fully employs the WebDAV protocol since it is a combination of a WebDAV server and an extension of an Internet browser to be used as WebDAV clients.

The documents in this system flow between clients and server. The addition of a new document to the server is represented by uploading the document via Put method. The processes of initiating and editing the documents are done locally. After locking and downloading the document using the appropriate methods, the user can edit the document on a local machine. After completing the document modification, he uploads the document to the server and unlocks it. While a document is locked by a user, another user can download and view it if he has the required privileges. However, he cannot download it for editing. The document repository is represented by the file system on the WebDAV server. But the files are arranged in three hierarchy levels: repository, folder and document levels.

2.13 MS OFFICE 2000 ANNOTATION SYSTEM

This system utilizes a new feature of Microsoft Office 2000, namely "Web Discussions", to provide a document annotation system (45). This feature allows

team members to make annotations to Web pages. The system is based upon client/server model. The client is a Web browser (the Internet explorer with an annotation client). On the other hand, there are two servers: a Web server and an annotation server. The annotation server contains a SQL Server database to maintain the annotations. While the Web server communicates with clients using HTTP protocol and contains the Web documents, the annotation server uses the WebDAV protocol and handles the document annotations. Figure 2.12 shows the system architecture.

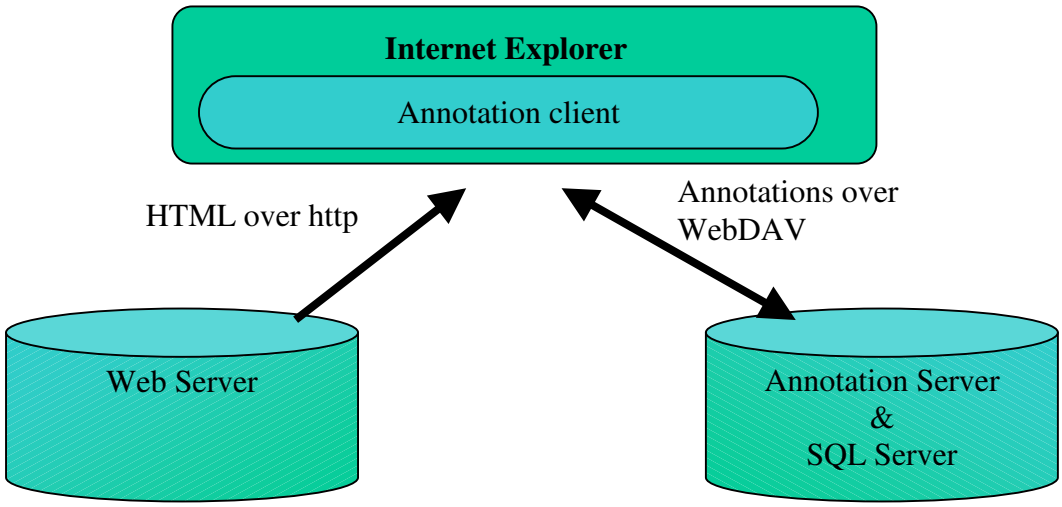


Figure 2.12: Office 2000 Annotation System

Using this system, the users can create annotations on any Web page in the Web server. Besides that, every author can edit or delete any of his annotations. There are two types of annotations: document annotations and paragraph annotations. Both of them need the document URL (i.e. the document address) while the latter requires also a unique signature for each paragraph. Moreover, the users

can subscribe to the notification system. The notification system informs each user about the annotation creation or updating of his subscribed documents.

There are some limitations of the system. One of these limitations is the technical orphaning of annotations, causing the system to fail in matching a paragraph annotation in a particular document to the correct location after updating the document. Another drawback is the absence of the access control. All annotations are viewable for everybody in the team.

2.14 FORM FLOW MODEL

This system is a protocol based system which converts the requirements of a document flow into an agent-based document flow system (46). This protocol represents a design methodology which consists of several steps: identifying its entities, identifying the system agents, building the agents, and implementing the system. This model handles just specific kind of documents which is forms. It stores the data of forms in a central database. Also, it uses KQML (Knowledge Query and Manipulation Language) to exchange information between the agents. Finally, in the implementation side, it uses JATLite which is a Java agent platform and presents the steps of the protocol for a course drop/add form.

2.15 X-FOLDERS

X-Folders is a system composed of one or more peer sites where each site contains several services and can invoke other services on other sites (47). The

services include copying documents, sending email messages, handling documents as web resources and so on. Thus, the application is distributed among sites. Moreover, each site is considered as a servnet (i.e. a server and a client at the same time).

The documents can either flow between the folders or use document status property. In the former case each folder represents a specific state. Moreover, the documents are stored in special folders which are accessible via WebDAV and SOAP.

Each folder contains documents and their properties including the status of the documents. Furthermore, each folder is mapped by a folder status tree. The folder status tree is an XML tree rooted at the folder and composed by subtrees which represent the documents stored in the folders. Each subtree of a document contains the properties of the documents.

Changing document status may fire specific tasks which consists of web services. X-Folders system takes the appropriate actions depending on the status of the folder. These actions are composed by web services. The tasks of X-Folders call the appropriate web services. The tasks are triggered after modifying the document repository.

2.16 CONCLUSION

This chapter includes summaries of fifteen different document flow and archival models. It is clear that several applications require document flow and archival facilities. Furthermore, the objectives of these applications as well as the

technologies available during the period of the application developments influence these models strongly. Consequently, the design of a structured frame of document flow and archival model becomes a necessary matter of fact. But this frame should be flexible so that different applications can utilize it.

CHAPTER 3

DFWDAV MODEL

This chapter contains a full description of the Document Flow model based on WebDAV protocol (DFWDAV) which we have developed. At the outset, the requirements of the model are identified. Then, the model components and architecture are tackled in details. After that, the model functionalities are elucidated. To clarify the roles of the model components and their operations, a scenario of a collaborative application is provided. By applying the DFWDAV model, the scenario shows the strength of the model. At the end of the chapter, the researcher will justify the fulfillment of the model requirements according to the provided scenario.

3.1 THE SPECIFICATIONS OF DFWDAV

While the model should be flexible to serve different document collaborative applications, its flexibility should not affect its intensity and capability. The foremost objective of the required architecture is to automate the routing of documents among distributed collaborating users. Thus, the architecture deals with cooperating users handling shared documents.

The model should be designed according to the following specifications:

1- **<u>Managing the production of documents:</u> The model should manage all the stages of documents from the creation to the production passing through the different phases of development. Usually, documents come across several users before they reach their final shape. During the flows of documents, the users either revise the documents or add more information to them. For example, reviewers of a book typically update it but do not add more information. Another different example, an employee vacation request flows between several responsible people who add new information to the request. In summary, the DFWDAV model should manage the documents during their flow between different users.**

2- **<u>Offering a document flow infrastructure:</u> To perform their tasks, most document collaborative applications need document flow functionality. On the other hand, the main purposes of these applications can be emphasized on if they are built on the top of a document flow infrastructure. Therefore, the DFWDAV model should provide a document flow and archival infrastructure for the collaborative applications as it is shown in Figure 3.1.**

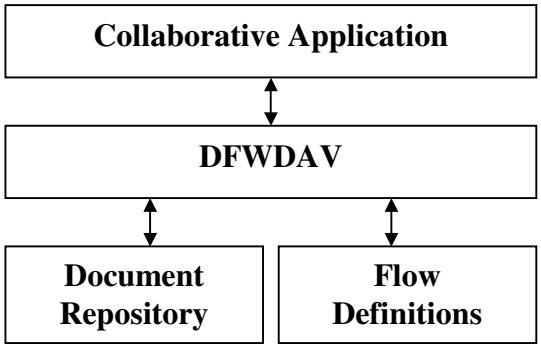


Figure 3.1: The DFWDAV Model as an Infrastructure for Collaborative Applications

Moreover, this feature simplifies the design and the production of such applications since they can utilize the DFWDAV model as a document flow infrastructure. For example, a collaborative authoring system can be built on the top of the developed document flow architecture. After an author initiates a document using the authoring system, he can send it to the developed architecture which handles the archive of the document. In the same manner, the developed architecture helps the collaborative system to lock, download and flow the documents. Therefore, collaborative applications can be more simplified as a result of utilizing the DFWDAV document flow model because they concentrate on their special businesses.

3- **Presenting the necessary security: One of the most important features of the DFWDAV model is its ability to share the documents between different authorized users according to their privileges. For example, an author initiates a document and other privileged users (e.g. reviewers) can check and may maintain that document. Moreover, those users should not violate their privileges on the document and unprivileged users should not access it.**

4- **Utilizing document repository: The DFWDAV model aims to utilize a document repository because such repository is the key of an archival system. The document repository can store and retrieve the documents and their metadata which includes the definition of the document flow, the user privileges and the document status.**

5- **Defining document flows: The DFWDAV model ought to provide an automatic document flow between users according to the document**

definitions. Therefore, privileged users should define the flow paths of documents in a structured way. Thus, the model manipulates the definitions of the document flows to automatically route the documents among the users.

6- **Providing automatic notification: An automatic notification is one of the most important features of the document collaborative systems. This feature allows the users to know when they should handle a document. Also, it notifies late users to push them to perform their work.**

7- **Using recent and open technologies: The DFWDAV model should utilize novel technologies. Moreover, it can be implemented without depending on a specific commercial product. In other words, it can be developed using open (i.e. noncommercial) products.**

8- **Supplying information about documents: Users need to know some information about documents during their development and after their production. This information includes current location and status in addition to the flow definition of a document.**

9- **Supporting the Internet environment: The Internet becomes an essential basis for contemporary applications since it provides a medium of communication between distributed users across the world. Consequently, the DFWDAV model should support the Internet in the best possible way.**

In summary, it is required to provide a document flow model which can represent an architecture of structured document flow systems that allow users on different sites to manage the creation and the production of documents in a well-

structured manner. Thus, the above specifications aim to emphasize the document flow functionality. So, several functions of document management systems are excluded from the model scope. Most of these functions are listed previously in subsection 1.1.3.

3.2 THE DFWDAV COMPONENTS

Despite the fact that the DFWDAV model utilizes the client/server architecture, it takes advantage of the Internet environment since it uses the WebDAV protocol. The model consists of two subsystems; the first is the server subsystem and the second is the client subsystem. Figure 3.2 shows the architecture of the model which contains the following components:

- **On the server subsystem:**
  - **WebDAV server.**
  - **Document repository.**
  - **Routing and reminding agent.**
  - **Flow repository.**
  - **Mail server.**
- **On the client subsystem:**
  - **WebDAV clients.**
  - **Document flow client.**
  - **Local temporary document repository.**

- **Non-WebDAV document editors.**
- **Flow definer.**
- **Mail client.**

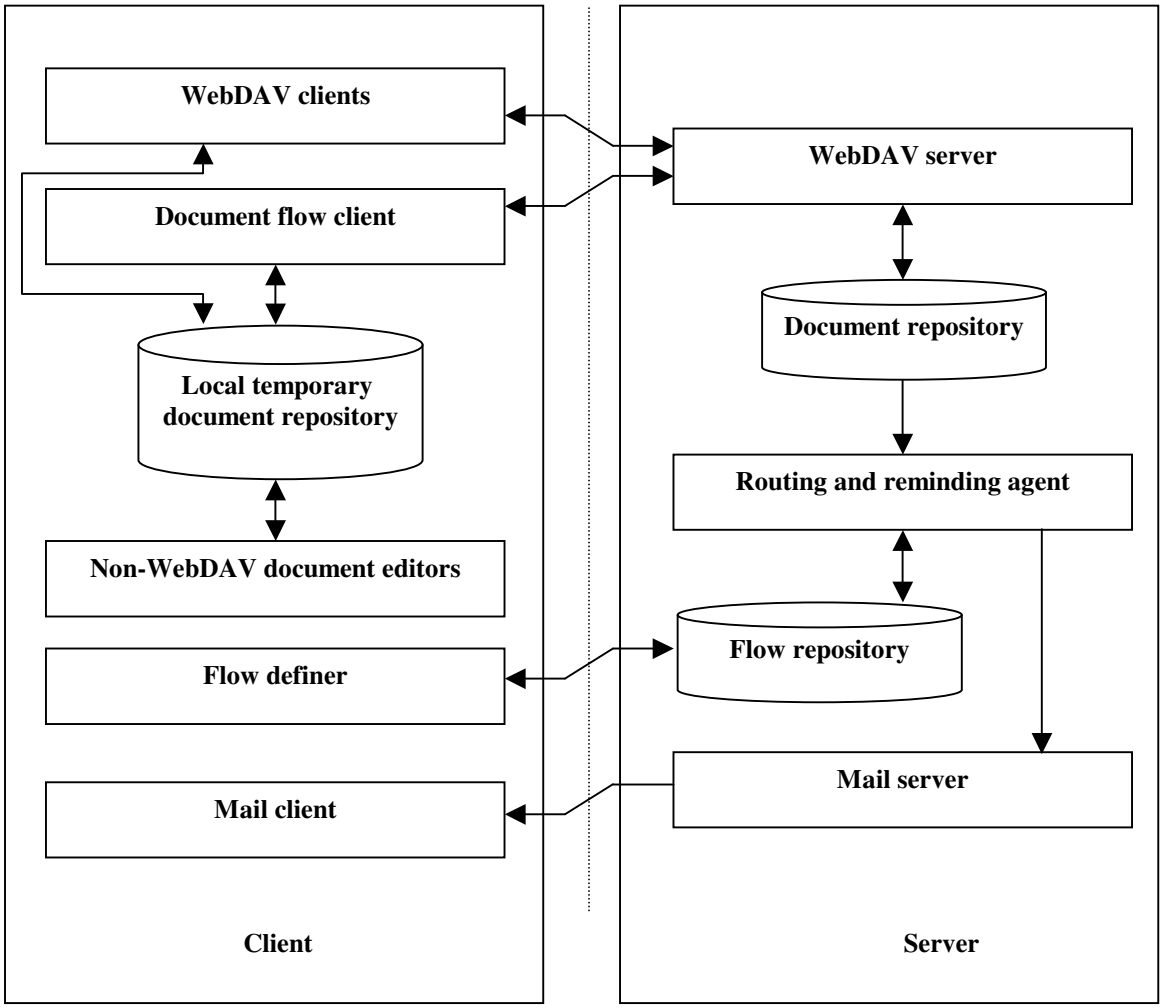


Figure 3.2: The Architecture for the DFWDAV

3.2.1 The Server Subsystem

As shown in Figure 3.2 the DFWDAV model is based mainly on the WebDAV protocol which is an extension of the HTTP protocol. This protocol will be

explained in detail in Chapter 4 and Appendix A. Also, the figure shows that the server subsystem includes a WebDAV server attached to a document repository, a special agent, a flow repository and a mail server.

The first component of the server is a WebDAV server. The WebDAV server, which is an extension of a web server, provides several functions to the DFWDAV model. These functions include:

- **Uploading documents from the clients to the server and downloading them from the server to the clients.**
- **Storing and retrieving the metadata (including the routing definitions) of documents to and from the repository.**
- **Defining the users' privileges for the documents.**
- **Locking and unlocking documents to prevent the lost-update problem.**
- **Providing the search capability for the document in the repository.**

The second component of the server is the document repository which holds the documents as well as their metadata. The metadata are divided to two parts: static metadata and dynamic metadata. The static metadata includes the attributes which are generated and modified by the model via the WebDAV server such as the author and the creation date. Moreover, the dynamic metadata contains also the status of the flowing documents in addition to the historical document flow information. In contrast, the dynamic metadata represents the document attributes that are defined and can be updated by the users such as the document name and the related document flow. According to the openness of the model, the repository

can be a database or a file system depending on the WebDAV server which is responsible for reading from and writing to the document repository.

The third component of the server is the routing and reminding agent which observes the user operations on the documents and their flow definitions. Then according to the operations and the flow definitions, it takes the appropriate actions. For example, if a user uploads a document to the server, which in turn should flow to a second user, the agent sends an email to the second user using the email server. This email asks him to download the document and to work with it. The functions of the agent will be tackled in detail in the next section (i.e. Section 3.3).

The fourth component of the server is the flow repository which holds the definitions of the flows. Each flow definition contains users who are involved in creating some documents as well as several properties of each user. Subsection 3.4.2 explains these properties comprehensively.

The last component of the server is the mail server, which receives the notifications from the routing and reminding agent via SMTP protocol. The server, in turns, sends the emails to the users. These emails notify the users about their turns in the document flows and remind them to download or upload the documents.

3.2.2 The Client Subsystem

The client of the DFWDAV model includes the following components: WebDAV clients, the document flow client, a local temporary document repository,

non-WebDAV document editors, the document flow definer and a mail client. The main component in the client subsystem is the WebDAV clients which are responsible for communicating with the WebDAV server. In spite of the abundance of WebDAV clients, they vary in their functionalities. Some WebDAV clients are just an extension of a web client and other WebDAV clients provide wide functionalities. For example, MS-Word is a WebDAV client while it is a well-known word processor.

The second component of the client is the document flow client. The users use it to download and upload documents between the document repository in the server subsystem and the local document repositories. Also, this component provides useful information to the users such as their responsibilities and notifications.

The third component in the client subsystem is the local document repository which can be used because of either two reasons. The primary reason is the non-WebDAV editors which can not edit documents directly on the server subsystem. The other reason is the unreliable communication media between the server subsystem and the client subsystem. In either case, users download documents locally where they edit them. After completing the modifications, users upload documents back to the document repository in the server subsystem.

The fourth component of the client is the non-WebDAV document editors. As shown earlier, some document editors (e.g. MS-Word) become WebDAV clients. But to generalize the model, the non-WebDAV document editors are considered as a

separate component. This also helps to make the model applicable to any type of documents and editors.

The fifth component of the client is the flow definer. This component plays a very important part of the model because users can use it to define the flow types of the documents between the users according to the organizational procedures and policies. A detailed explanation of this component and its responsibilities will be handled in the next section.

The final component of the client is the mail client which is part of the mailing system. While the routing and reminding agent in the server subsystem sends notifications to the server, mail clients deliver these notifications to the users.

3.3 THE FUNCTIONS OF DFWDAV

This section provides an overview of the major functions of the model. These functions include creating the documents, updating the documents, defining the document flows, defining and querying the document metadata, routing the documents, notifying the users and archiving the documents.

3.3.1 Creating the Documents

Using the document editors, users can create new documents which are usually initiated in the client subsystem. Then, their initiators upload them with their initial metadata to the server. The initial metadata includes the definition of

the flow path. Moreover, some WebDAV clients such as MS-Office can create the documents directly in the server.

3.3.2 Updating the Documents

There are two ways to update documents. The first way is by using the WebDAV clients in which users can edit documents directly on the server subsystem. The other way is to update the documents locally after downloading them from the server subsystem to the local document repositories in the client subsystem. A user can only update a document if it is his turn in the definition of the document flow. When he downloads it for updating, it will be locked on the server until he finishes the modifications and uploads it back to the server.

3.3.3 Defining the Document Flows

It is possible to define the flow of a document by using the document flow definer. The definition of the flow is sent to the server as one of the metadata (i.e. the properties) of the document. On the server, the routing and reminding agent analyzes the flow definition and then takes the appropriate actions. The flow definition includes the users who are supposed to handle the document and their work durations. More information about such definitions appears in Section 3.4.

3.3.4 Defining and Querying the Document Metadata

The metadata of a document is the data which represents the document properties such as the creator of the document, its current location and the related flow definition. The metadata can be divided to two parts. The first part includes static properties which are automatically defined and only changed by the system. An example of such properties is the date of the creation. The second part is the dynamic properties which can be defined and changed by users if they have the sufficient privileges. The access privileges of a document and its related flow can be considered as dynamic properties.

3.3.5 Routing the Documents

One major role of the routing and reminding agent is to analyze the definitions of the flow paths. Accordingly, the agent takes the appropriate actions which include emailing the users to download the documents and skipping them if they do not finish their work on the determined periods.

3.3.6 Notifying the Users

The second major role of the routing and reminding agent is the user notifications which provide the users by suitable information. This information includes the actions which they should do as well as the current status of documents. For example, the agent notifies a user to download a document when it is his turn in

the flow path. Another example, it reminds a late user to upload the document back to the server.

3.3.7 Archiving the Documents

When the server receives a document from a client, it sends the document to the document repository where the document will be archived. In the same manner, the metadata will be saved in the repository which can be either a database or a file system. Following the clients' requests, the server retrieves the documents and their metadata from the repository and sends them back to the clients.

3.4 UTILIZING DFWDAV

This section aims to clarify the work method of the DFWDAV model as well as the functions of the model and the relationships between the model components. It begins by browsing several possible applications that can employ the model. Then, a full scenario of an authoring application is discussed in detail to show the capability of the model.

3.4.1 Applications

As in the specification section, the DFWDAV model can be considered as an infrastructure of document collaborative applications. Although there are several

collaborative applications, the following applications can be considered as the most common ones:

- **<u>Collaborative authoring:</u> Collaborative authoring application is the activities involved in the production of a document by more than one author. So, the process of this application leads to different publications such as books, researches and articles. In a regular manner, several users participate in the collaborative authoring chronologically. After the authors write the documents, the reviewers revise them and then the editors arrange them to produce the required publications.**
- **<u>Request handling:</u> A request is a kind of a document in which several users add some information. Therefore, usually there is no updating on the requests. For example, an employee can initiate a vacation request. This request passes through a specific path of users. Each user in the path appends some information to the request until it reaches its final stage.**
- **<u>Distributing documents between the users:</u> In several real applications, the objective of the documents is to distribute some information to particular users. For example, distributing instructions and announcements is a daily process in our life. In this application, the documents flow from the initiators to the readers.**

3.4.2 Scenario

To clarify the relationship between the components of the model and how they work, we will describe how the model can be utilized as an infrastructure for an authoring application.

Usually, an authoring system is operated by several authors, reviewers, and editors. Let us discuss how can these users produce a book using the DFWDAV model. First, the book should be divided into several parts according to the roles of the users. Initially, each part can be considered as a separate document. This consideration increases the system productivity, since it ensures the parallel work of the users. Therefore, selecting a suitable document granularity helps the users to work simultaneously. As a result, rather than assuming that the whole book as one document, it is possible to divide it into several documents by selecting a very delicate granularity.

For the purpose of this section, we will assume that the scenario will be according to Table 3.1. Therefore, each chapter can be considered as the document granularity since it is written by one author and then will be reviewed serially by several reviewers. Thus, each author can initiate his own document (i.e. chapter).

Chapter	Author	Reviewers
1	Ali	Ahmed, Saad and Selah
2	Mohammed	Saad, Ahmed and Ali
3	Selah	Ahmed, Saad and Ali
4	Ahmed	Saad, Ali and Selah

Table 3.1: Work Distribution of the Authoring Scenario.

Also, Table 3.1 provides the flow definition of each chapter. As it appears above, several users will review each chapter. For example, Ahmed, Saad and Selah should review chapter 1 consequently. Moreover, for each person in the flow path, there are several properties. Table 3.2 shows an example of the flow properties of the first chapter.

Property	**Ahmed**	**Saad**	**Selah**
Number of downloading notifications	**3 times**	**4 times**	**4 times**
Downloading notification period	**1 day**	**½ day**	**½ day**
Number of uploading notifications	**3 times**	**6 times**	**6 times**
Uploading notification period	**1 day**	**½ day**	**½ day**
Working period	**7 days**	**5 days**	**3 days**

Table 3.2: The Flow Properties of Chapter 1 in the Scenario.

There are five properties; the first two of them are related to the downloading, the third and fourth are related to the uploading and the fifth represents the working period, that is the period between downloading a document and sending the first uploading notification. The properties of the downloading are the number of downloading notifications and the downloading notification period. The download notifications recommend the users to download the documents from the server and to perform their work with the documents. The number of downloading notifications of a user is the number of notifications which the agent should send to the user to remind him to download the document. If the user does

not download the document before the end of the notifications, the agent will skip him. The second property is the download notification period which represents the period between two download notifications.

Similarly, there are two properties related to the uploading. These properties are the number of uploading notifications and the uploading notification period. An upload notification will be sent to the user if he does not upload the document after the working period finishes. The upload notification requires him to end his work on the document and to upload it back to the server. If the user does not upload the document after the first notification, he will receive another notification after the uploading notification period. When he does not upload the document and the uploading notifications reach the number of uploading notifications, the agent will skip him. Figure 3.3 shows the processes of analyzing the document flows by the routing and reminding agent. Finally, after reviewing all chapters, the editor is responsible to combine them together to produce one document which represents the book.

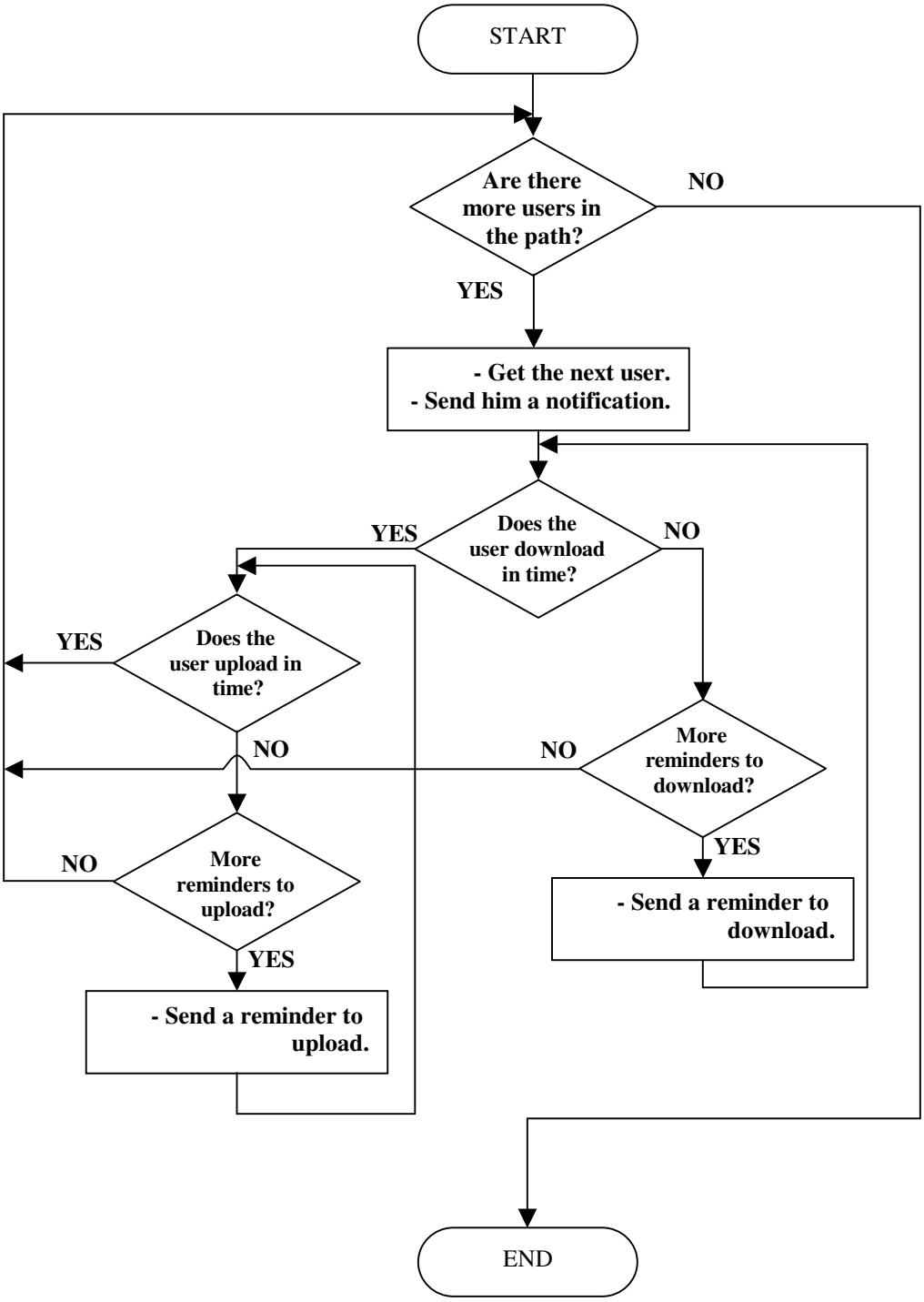


Figure 3.3: Analyzing the Flows by the Routing and Reminding Agent.

3.5 THE FULFILLMENT OF THE SPECIFICATIONS

In this section I will verify the fulfillment of the model to the specifications listed in section (3.1). These specifications were:

1. **Managing the production of the documents: Several functions of the DFWDAV model cooperate to manage the production of the documents. These functions start by document creation and end by document archiving. Moreover, these functions include defining document flow paths, updating documents and routing them.**
2. **Offering a document flow infrastructure: As shown in section (3.4), several types of applications can utilize the DFWDAV model. Furthermore, the discussed scenario shows how an authoring system can be built on the top of the DFWDAV model.**
3. **Presenting the necessary security: The DFWDAV model applies WebDAV protocol. This protocol provides a strong security as will be shown in the next chapter. The implementation shows different levels of security. For example, it is possible to define the allowed operations on the document level for each user. Also, only the current user in the flow path can update the document.**
4. **Utilizing document repository: As previously explained in the model components, the document repository is an important component which holds the documents and their properties. The server is responsible for sending and**

receiving the documents to and from the server. Since the document repository is a standalone component, it is possible to use a database or a file system as the document repository. Consequently, the model provides a wide flexibility of choosing a document repository.

5. **Defining document routing (i.e. flows): Defining the document flow paths is the starting point of the document flows. The model allows the user to define the flow paths when they have the sufficient privileges. The defined flow paths in turn can be assigned to the documents. The agent analyzes the assigned flow paths and accordingly takes the appropriate actions.**
6. **Providing automatic notification: The agent is responsible for sending the notifications. There are several types of notifications, for example a user of a document flow receives the notification of his order. This notification informs him about his turn in the document flow and requests him to download the document. Another example is the download notifications which also request him to download the document when he does not do so in time. In contrast, the upload notifications request the user to upload the document when his working period is over. The last type of notifications is to inform the users of canceling them from the flow when they do not download or upload the documents in the specified periods.**
7. **Using recent and open technologies: The model utilizes WebDAV which is a recent emerging network protocol. This protocol hides the bulk of the complexity of the application implementation. Moreover, the model is open since it does not require a specific document type and since it can work with any**

document type. Wide possible implementations of the model are available as will be presented in Chapter 5. Some of the implementations rely completely on the open (i.e. noncommercial) products.

8. **Supplying information about documents: The model provides several kinds of information related to a document, for example it is possible to browse the flow path of the document. Also, the model provides history information about the document operations of the users and about the notifications sent. Moreover, since the DFWDAV model utilizes the WebDAV protocol, it supports complex types of document properties.**
9. **Supporting the Internet environment: The model supports the Internet because it is built on the top of the WebDAV protocol. The WebDAV protocol is based on the HTTP protocol, which is a well-known protocol in the Internet environment. The implementation which is presented in Chapter 5 shows that the model widely supports the Internet work.**

CHAPTER 4

WEBDAV AS AN INFRASTRUCTURE

The objective of this chapter is to show how WebDAV, which is introduced in Chapter 1, can implement the infrastructure of the collaborative applications such as the DFWDAV. First, this chapter provides the requirements of Web-based collaborative systems and highlights the problems faced by such systems using HTTP protocol. Second, the objectives and the workgroups of the WebDAV are listed. Finally, the WebDAV functions and methods are briefly explained.

4.1 THE REQUISITES OF DISTRIBUTED WEB COOPERATIVE APPLICATIONS

The Web-enabled distributed collaborative systems involve diverse requisites. Those requisites involve locking, security, properties, reservations, versioning and collections (48).

- Locking mechanisms to prevent overwrite conflicts:

Locking mechanisms prevent the lost update problem. In other words, they prevent more than one user from modifying a document simultaneously

and ensure that only one person may modify it. A lock can be on multiple documents but it should occur at the same action across the multiple documents (i.e. the locking operation must be atomic throughout these documents). This is vital to prohibit a collision between several people trying to establish locks on the same set of documents. Removing a lock is an obvious operation. Other locking operations include determining the active locks and finding out who holds those locks.

- Security:

Several security aspects are compulsory obligated for distributed document management systems. Those include authorization, access rights and authentication. A document must be accessed by authorized people only. If a person has no authorization on a specific document, he should not be able to access it. Each authorized person should have defined access privileges. Access privileges can be browsing, editing, and copying, etc. It is possible to restrict modification of a document to a specific person. Authentication ensures the identity of the users, the integrity of the messages and the privacy of communication.

- Document Properties:

Document properties provide descriptive information about documents. They include bibliographic information such as author, title, publisher, and subject, etc. These properties can be used for supporting

searches on property values and enforcing copyrights. The properties operations include creating, modifying, reading and deleting arbitrary document properties.

- Reservations:

A reservation is a declaration that one aims to edit a resource. In the server of a given resource, a person can register an intent to the resource, so that other users can discover who intends to edit the resource. Reservation operations include determining the active reservations of a given resource, finding who currently holds reservations and releasing a reservation.

- Versioning

With versioning, the history and the evolution of a document can be provided accurately. The versioning operations include many tasks such as: reserve existing version, lock existing version, retrieve existing version, request or suggest identifier for new version, write new version, release lock, release reservation. Versioning systems combine reservation, locking, and retrieval into an atomic checkout function or some subset of these tasks. They combine writing, releasing the lock, and releasing the reservation into an atomic checkin function or some subset of these tasks.

Each version typically stands in a "derived from" relationship to its antecessor(s). It is possible to produce several different versions out of a single version which is called branching. Also, it is possible to produce a

single version from several versions that is merging. Consequently, the collection of related versions forms a directed acyclic graph which is called a "version graph". Each node of this graph is a "version". The edges of the graph denote the "derived from" relationships. Version graph operations include referring to a version graph, referring to a specific member of a version graph, determining the version graph of a given document if it is available, navigating a version graph from a given node (document) to the related members, and providing information about a version graph.

- Collections:

A collection is a resource that contains other resources. Collections are useful for arranging documents. The collection operations include creating, copying, deleting and reading the content of a collection.

4.2 THE PROBLEMS FACED BY WEB COLLABORATIVE SYSTEMS USING HTTP PROTOCOL

While the current functionality of the Hypertext Transfer Protocol (HTTP/1.1) enables the editing of the Web content at a remote location in addition to remote browsing of content, it contains a limited support for remote collaborative applications. Consequently, HTTP does not provide sufficient functionality for collaborative Web applications. Distributed Web systems that use HTTP suffer

from potential problems (25), (48). Some of those problems are listed in the following:

- **HTTP provides limited support for preventing a user from overwriting other users' modifications. In other words, using HTTP by distributed Web applications can lead to the "lost update problem" or the "overwrite problem". Therefore, it is inadequate to support efficient remote editing free of overwriting conflicts.**
- **Using HTTP, there is no way to discover whether someone else is currently making modifications to a certain document.**
- **HTTP headers include the parameter information of methods. Consequently, the length of parameters is bounded and so there is no direct way to pass variable length parameters.**
- **HTTP does not support a sufficient level of security.**
- **The HTTP protocol contains functionality which enables the editing of Web content at a remote location, without direct access to the storage media via an operating system.**

As a result of the previous mentioned points, the HTML applications and tools have shown inability to meet the needs of their users while using the facilities of the HTTP protocol. So, several Web collaborative systems try to overcome those problems by adding extensions to the HTTP protocol. But since those extensions were developed separately, they were not interoperable. Accordingly, the need to provide a standard extension is an obvious matter.

4.3 WEBDAV OBJECTIVE AND GROUPS

The objective of the main WebDAV group is to define the WebDAV protocol, which provides the following functionality:

- **Locking: lock, lock status, unlock.**
- **Name space manipulation: copy, move/rename, and resource redirection.**
- **Containers: creation, access, modification, and container-specific semantics.**
- **Attributes: creation, access, modification, query and naming.**
- **Notification of intent to edit: reserve, reservation status and release reservation.**
- **Use of existing authentication schemes.**
- **Access control.**
- **Versioning: checkin/checkout, history graph, differencing, automatic merging, and accessing resource versions.**

While the WebDAV group handles the generic issues of the WebDAV as well as the base level of the WebDAV standards, several related groups and sub-groups are created to investigate specific problems and sustain the original group (49), (50). Those groups include:

- **DASL (DAV Searching and Locating): This working group is functioning on searching facilities for DAV repositories.**
- **Delta-V (Web Versioning and Configuration Management): WebDAV was not able to finish versioning, so Delta-V is picking up where WebDAV group**

is left off. Thus, Delta-V group is responsible to extend the Web with versioning and configuration management support.

- **Access Control: This is a sub-working group of WebDAV, concerned with developing a remote access control protocol.**

These groups produced massive documentation related to their problems. Consequently, we will select a subset of this documentation. This subset provides the basic functionality required by the DFWDAV. Table 4.1 lists the selected documentation.

This documentation extends HTTP/1.1 methods and headers. In addition, it specifies how to use the new extensions, how to format request and response bodies and how existing HTTP behavior may change.

Workgroup	Document	Current Status	Date
WebDAV	HTTP Extensions for Distributed Authoring (24)	RFC2518	Feb. 1999
Delta-V	Versioning Extensions to WebDAV (51)	RFC3253	Mar. 2002
Access Control	WebDAV Access Control Protocol (52)	RFC3744	May 2004
DASL	WebDAV Search (53)	Internet Draft	Feb. 2004

Table 4.1: WebDAV Groups and Their Documents.

The next section summarizes most of the methods of the previous documentation which is needed to provide the DFWDAV. In this section, the full structures of the requests and responses are ignored. Meanwhile Appendix A provides several examples of them.

For more information about WebDAV, one should consult www.webdav.org site. It was created in February 1999 to provide the WebDAV community with a central location for "All Things WebDAV." The site provides pointers to many types of information and materials, along with pointers to WebDAV software (54).

4.4 WEBDAV METHODS

The WebDAV protocol overcomes the HTTP protocol by means of providing assorted methods which support the collaborative mechanisms. This section is dedicated to most of the WebDAV methods divided according to the WebDAV documents as shown in Table 4.1.

4.4.1 HTTP Extensions for Distributed Authoring

There are several essential WebDAV protocol methods that allow distributed clients to perform remote web content authoring operations (24). This subsection summarizes the methods of the properties, collections, namespace operations and locks.

Properties:

A property is a name/value pair that contains descriptive information about any resource. There are two types of properties: live and dead. A live property is a property whose semantics and syntax are enforced by the server. A dead property is a property whose semantics and syntax are not enforced by the server. Since the types of documents that people use vary, the list of possible property types becomes infinite. XML is the type of extensible communication vehicle which is required by WebDAV. XML properties provide storage for arbitrary metadata, such as a list of authors on Web resources. These properties can be efficiently set, deleted, and retrieved using the DAV protocol. Table 4.2 presents the methods of the properties.

Method	**Description**	**Input**
PropFind	**Retrieves properties defined on a Resource or a Collection with its internal members (depend on Depth).** **It is possible to request particular property values, all property values, or a list of the names of the resource's properties by setting an XML element.**	**Resource** (URI) or **Collection** **Depth** (0,1 or infinity) **XML element**
PropPatch	**Sets and/or removes properties on a resources or a collection**	**Resource** (URI) **or Collection** **XML element**

Table 4.2: The Methods of the Properties

Collections and Namespace Operations:

A collection is a resource that contains a set of URIs. WebDAV introduces the notion of the collection (analogous to a file system folder), which can contain resources. It is possible to create and list the collections like the directories in the file systems. Namespace management provides the ability to copy and move resources, and to receive a listing of resources at a particular hierarchy level (like a directory listing in a file system). Since resources may need to be copied or moved as a Web site evolves, DAV supports copy and move operations. Table 4.3 shows that WebDAV includes the ability to create, move, copy, and delete collections, as well as the ability to do the same things to the resources or files within the collection.

Locks:

Using a lock, an authoring client can provide a reasonable guarantee that another principal will not modify a resource while it is being edited. In this way, a client can prevent the "lost update" problem in which modifications are lost as first one author, and then another writes their changes without merging the other author's changes. Therefore, Lock ensures the ability to keep more than one person away from working on a document at the same time.

A lock token is returned by every successful LOCK operation. A lock token is a type of state token, represented as a URI, which identifies a particular lock.

Method	Description	Input
MKCOL	**Create a new collection.**	**URI**
DELETE	**DELETE for Collections:** **Removes the collection specified in the Request-URI and all resources identified by its internal member URIs.** **DELETE for Non-Collection Resources:** **Removes the Request-URI from all collections containing that URI as a member.**	**URI**
GET	**Retrieves whatever information is identified by the Request-URI in the form of an entity.**	**URI**
POST	**The semantics of POST are unmodified.**	**URI, Entity**
PUT	**Stores the enclosed entity under the supplied Request-URI.**	**URI** **Entity**
COPY	**Creates a duplicate of the source resource in the destination resource.** **If a resource exists at the destination and the Overwrite header is "T" then prior to performing the copy the server MUST perform a DELETE with "Depth: infinity" on the destination resource. If the Overwrite header is set to "F" then the operation will fail.**	**SourceURI** **DestinatioURI** **Overwrite**(T/F) **Depth** (0,1 or infinity)
MOVE	**The MOVE operation on a resource is the logical equivalent of a copy (COPY), followed by a consistent maintenance processing, followed by a delete of the source, where all three actions are performed atomically.**	**SourceURI** **DestinatioURI** **Overwrite**(T/F)

Table 4.3: Collections and Namespace Methods

To achieve robust Internet-scale collaboration, where network connections may be disconnected arbitrarily, and for scalability, since each open connection consumes server resources, the duration of DAV locks are independent of any individual network connection. Table 4.4 summarizes the lock methods.

Method	Description	Input
LOCK	**Takes out a lock and returns a lock taken. A lock on a resource must also lock the properties of the resources.** **If the Depth header is set to infinity then the resource specified in the Request-URI along with all its internal members in its hierarchy are to be locked.**	**URI** **Timeout** **Depth** (0,1 or infinity)
UNLOCK	**Removes the lock identified by the lock token from the Request-URI.**	**URI** lock token

Table 4.4: Locking Methods

4.4.2 Versioning Extensions to WebDAV

Version management provides the ability to store important revisions of a document for later retrieval. Version management can also support collaboration by allowing two or more authors to work on the same document in parallel tracks.

WebDAV Versioning defines two levels of versioning functionality: basic versioning and advanced versioning (51). Basic versioning provides versioning of largely independent resources. It allows authors to concurrently create, label, and access distinct revisions of a resource, and provides automatic versioning for versioning-unaware clients. Advanced versioning provides more sophisticated capabilities such as logical change tracking, workspace management, and versioning the URL namespace. Table 4.5 lists only the basic versioning methods since they provide what the DFWDAV needs.

4.4.3 WebDAV Access Control Protocol

Several requests provide the ability to set and clear access control lists (ACLs) which are associated with resources (52). An ACL contains a set of "access control entries" (ACEs), where each ACE specifies a principal and a set of rights that are either granted or denied to that principal. A principal can be a user, a client software, a server or a group. This functionality is crucial for allowing collaborators to remotely add and remove people from the list of collaborators on a single resource which is shown in Table 4.6.

4.4.4 WebDAV Search

It is required to search for DAV resources based upon a set of client-supplied criteria. This will minimize the complexity of clients so as to facilitate widespread

deployment of applications which are capable of utilizing the WebDAV search mechanisms (53). The WebDAV Search method is described in Table 4.7.

Method	Description	Input
VERSION-CONTROL	**Creates a version-controlled resource at the request-URL**	**URL**
CHECKOUT	**A CHECKOUT request can be applied to a checked-in version-controlled resource to allow modifications to the content and dead properties of that version-controlled resource.**	**URL**
CHECKIN	**A CHECKIN request can be applied to a checked-out version-controlled resource to produce a new version whose content and dead properties are copied from the checked-out resource.**	**URL**
UNCHECKOUT	**An UNCHECKOUT request can be applied to a checked-out version-controlled resource to cancel the CHECKOUT and restore the pre-CHECKOUT state of the version-controlled resource.**	**URL**
REPORT	**A REPORT request is an extensible mechanism for obtaining information about resources.**	**URL** **Report type**

Table 4.5: Versioning Methods

Method	Description	Input
ACL	**Provides a mechanism to set ACL information. Its request body is used to define alternatives to the ACL of the resource identified by the request-URI.**	**URI** ACLinformation

Table 4.6: Access Control Methods

Method	Description	Input
SEARCH	**Initiate a server-side search. It does not define the semantics of the query. In contrast, the type of the query defines the semantics.**	**URI** Query

Table 4.7: Search Method

The bulk of the WebDAV Search workgroup work is related to query grammar. One can express the basic usage of the WebDAV search in the following steps:

- **The client constructs a query using the DAV:basicsearch grammar.**
- **The client invokes the SEARCH method on a resource that will perform the search (the search arbiter) and includes a text/xml request entity that contains the query.**
- **The search arbiter performs the query.**

- **The search arbiter sends the results of the query back to the client in the response. The server sends a text/xml entity that matches the PROPFIND response.**

CHAPTER 5

IMPLEMENTATION

This chapter provides an authoring system implementation based on the DFWDAV model. At the outset, it shows that the computing environment provides widespread support to the WebDAV protocol which represents the basis of the model. Moreover, several of the famous WebDAV software products, which are used in the implementation, are browsed. Next, the chapter contains the implementation of a collaborative authoring system based on the DFWDAV model.

5.1 WEBDAV PRODUCTS

While WebDAV protocol is still evolving, it is widely supported by the computing environment. Several software products provide most of the WebDAV methods' functionality. Moreover, several Internet sites act as WebDAV servers and provide WebDAV account to the public (54). In addition, several of WebDAV software products are open source software and some of them provide API (Application Program Interface) for WebDAV programmers.

WebDAV software products can be divided to WebDAV servers and WebDAV clients. The WebDAV servers include Apache HTTP Server, Jigsaw, Slide, Microsoft Exchange Server, Microsoft Internet Information Services (IIS), Oracle 9i Enterprise Edition, Oracle Internet File System (IFS), SAP Portals Enterprise Portal Server, Tamino WebDAV Server and Xythos WebFile Server. Some of these servers are available to the public on the Internet as either test sites or document repositories.

The Apache HTTP Server is a famous web server which is becoming the most widespread web server in the world (55). The Apache Group is responsible about managing and developing this server. The mod_dav module is the Apache module which provides the WebDAV functions. On the other hand, Jigsaw is also a web server but it is developed by Java language and is sponsored by W^3C (56). It provides several interesting features including the support of WebDAV protocol. Also, Slide is a content management project based on WebDAV using the Java language (57). While it is still under construction, it is expected to consist of several modules such as a content management system, a WebDAV client library, a WebDAV command line client and a WebDAV server. Moreover, it provides Java API which can be utilized by programmers to developed WebDAV applications.

Several WebDAV servers can be tested directly on the Internet because they are installed on test sites which can provide free accounts for anybody and can be used for testing WebDAV protocol and WebDAV clients. For example, SAP Portals Enterprise Portal Server, Tamino WebDAV Server and Xythos WebFile Server are

available on public sites which support WebDAV protocol as will be shown in the next section.

On the other hand, the WebDAV clients are increasing rapidly. They include DAV Explorer, Microsoft Web Folders, Microsoft Office, Microsoft Internet Explore, Tamino WebDAV Basic Client, XML Spy, jEdit, SkunkDAV and IBM WebSphere DAV for Java (DAV4J). As an example of WebDAV clients, the DAV Explorer is a WebDAV client developed by Java (58). The most fantastic feature of DAV Explorer is its graphical user interface which is similar to the Explorer program in the Windows operating system. In the next section, more information about other clients will be given.

Furthermore plenty of open source software products support the WebDAV protocol. Open source software products are software products that are not developed for commercial benefits and whose source is available for anybody. Apache HTTP Server, Jigsaw, Slide and DAV Explorer are some examples of the open source software products that are WebDAV products.

In addition, some of WebDAV products provide WebDAV libraries as Application Program Interface (API) which can be used by the programmers to develop WebDAV applications. Some examples of such products are Slide, SkunkDAV and DAV4J.

5.2 IMPLEMENTATION OF A COLLABORATIVE AUTHORING SYSTEM

In this section, I present a collaborative authoring system implementation as a proof of the concept. This authoring system is based upon the DFWDAV model which is explained in depth previously in Chapter 3. The system enables distributed users to produce publications in a smooth and efficient way. The objectives of this implementation include clarifying duties of authors and guaranteeing standardized works.

As the model is based upon the client/server paradigm, the system consists of two main parts as shown in Figure 5.1. The first part is the server which contains the WebDAV server, the document repository, the routing and reminding agent, the

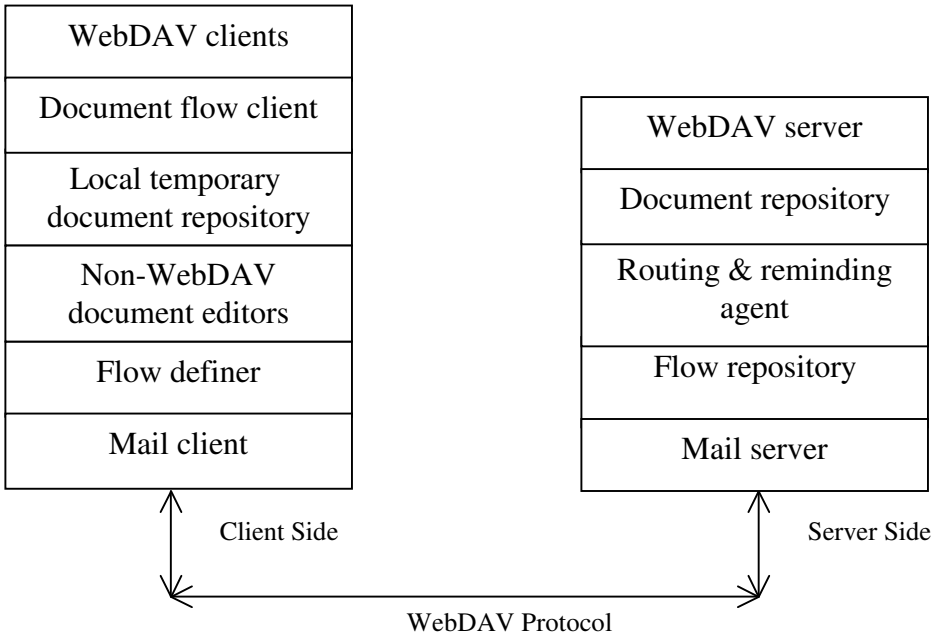


Figure 5.1: Implementation Components

flow repository, and the mail server. The second part is the client which consists of WebDAV clients, the document flow client, the local temporary document repository, non-WebDAV document editors, the flow definer and the mail client. The following sections describe the system parts in detail.

5.3 THE SERVER SUBSYSTEM

The server subsystem is the first part of the system. This subsystem includes the WebDAV server, the document repository, the routing and reminding agent, the flow repository, and the mail server.

5.3.1 The WebDAV Server

The WebDAV server is responsible to get the requests from WebDAV clients to handle them. According to the type of the requests, the server takes the appropriate actions. Most of the requests are illustrated comprehensively in Appendix A. For example, if a client sends a lock request for a specific document, the server checks the user privileges on that document as well as its current state and if there are no obstacles it locks the document for the user. Also, the WebDAV server handles the document repository which holds the documents.

In the implementation, I tested several WebDAV servers including Apache HTTP Server, Tamino WebDAV Server, SAP Portals Enterprise Portal Server, Xythos WebFile Server and Oracle 9i Enterprise Edition. There are several reasons for testing several servers, such as showing the strength and the capability of the

model. Another reason is the ability to test as most as possible of WebDAV methods because some servers do not implement some WebDAV methods yet. For example, Apache HTTP Server implements only most of the core methods of the WebDAV protocol whereas Tamino WebDAV Server and Xythos WebFile Server implement some of the versioning and access controlling methods. Concisely, the description of each server, the way of using it as well as its demonstrated characteristics will be elaborated in the following paragraphs.

- **Apache HTTP Server**

  I locally installed and configured the Apache HTTP Server, which is an open-source product. The local installment of the server allows it to communicate with the routing and reminding agent. Unfortunately, the current last version of Apache HTTP Server implements only the core WebDAV methods although the next versions of it may implement other methods. After installation, the Apache HTTP Server needs some configuration to support WebDAV. The minimal configuration includes the following steps:

  1- **Enabling the WebDAV functionality by loading its module by adding the next lines to the configuration file:**

  ```
  LoadModule dav_module modules/mod_dav.so
  LoadModule dav_fs_module modules/mod_dav_fs.so
  ```

  2- **Specifying a web-server writable filename as the WebDAV lock database by adding the following to the configuration file:**

  ```
  DavLockDB {FILENAME}
  ```

3- **Adding the DAV directive in the WebDAV repository container in the configuration file to enable the WebDAV methods, for example:**

```
Alias /repository/ {PATHNAME}
<Directory {PATHNAME}>
    DAV On
</Directory>
```

4- **Restarting the Apache HTTP Server.**

After configuring and restarting the server, it is possible to check the result in the Apache Service Monitor screen. As shown in the bottom of the screen in Figure 5.2, the server started with WebDAV support. The WebDAV repository on Apache HTTP Server becomes http://localhost/repository/ .

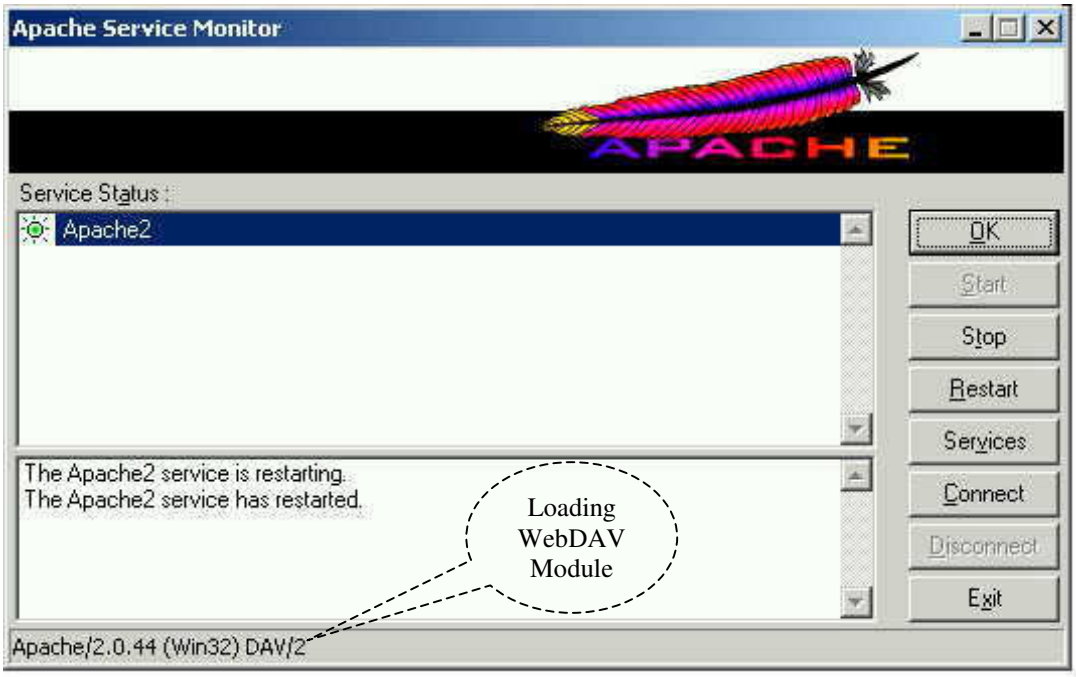


Figure 5.2: Apache Service Monitor Screen after Loading WebDAV Module

- **Tamino WebDAV Server**

Since Apache HTTP does not yet provide versioning support, I tested several other servers directly on the Internet because they are commercial products and they have copyrights. But working on the Internet limits the connection with the agent. For example, Tamino WebDAV Server is one of these servers which are installed in public sites. To test the Tamino Server, I requested a testing account. Then, I received a WebDAV URL and a series of user and password combinations and I became a registered user. My URL was http://tamino.demozone.softwareag.com/taminowebdavserver/gemini/. This URL can be accessed using web browsers with limited access as will be discussed in WebDAV client subsection and as appears in Figure 5.3.

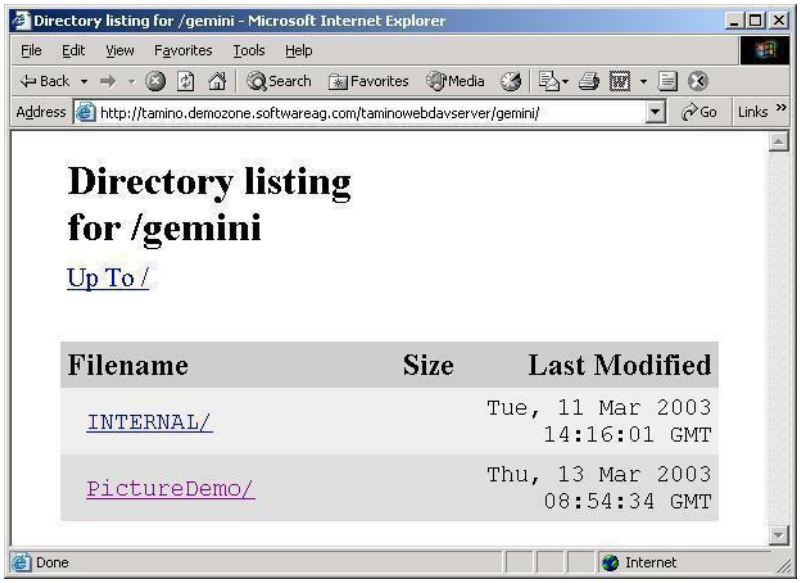


Figure 5.3: Accessing Tamino WebDAV Server From a Web Browser

- **SAP Portals Enterprise Portal Server**

SAP Portals Enterprise Portal Server is another server which implements WebDAV versioning methods which I tested on the Internet. The address of my repository on SAP Portals Enterprise Portal Server page is http://greenbytes.de:81/wcm/docs/cm/u39/. Several functions of this server, such as versioning and locking, can be accomplished by using web browsers as presented in Figure 5.4 and in the WebDAV client section.

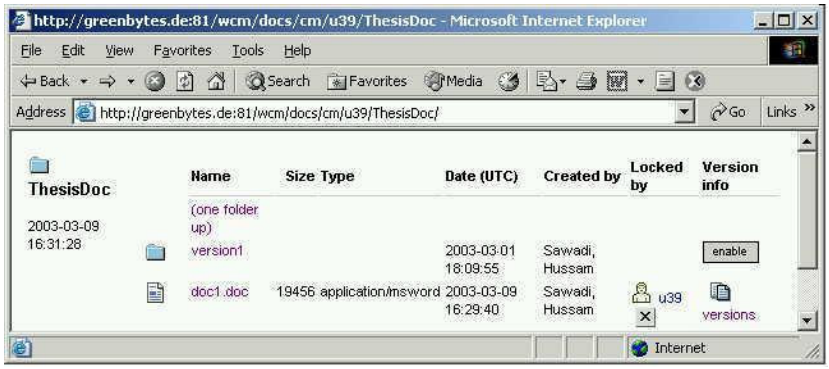


Figure 5.4: Accessing SAP Portals from a Web Browser

- **Xythos WebFile Server**

Xythos WebFile Server is a WebDAV server which I tested on the Internet. Fortunately, the Xythos WebFile Server on the Internet provides a fruitful site. The name of this site is Sharemation which can provide its full functionality through a web browser with a friendly graphical user interface.

Figure 5.5 shows the Internet Explorer Browser accessing the URL of my account on this site which is http://www.sharemation.com/wellxml/.

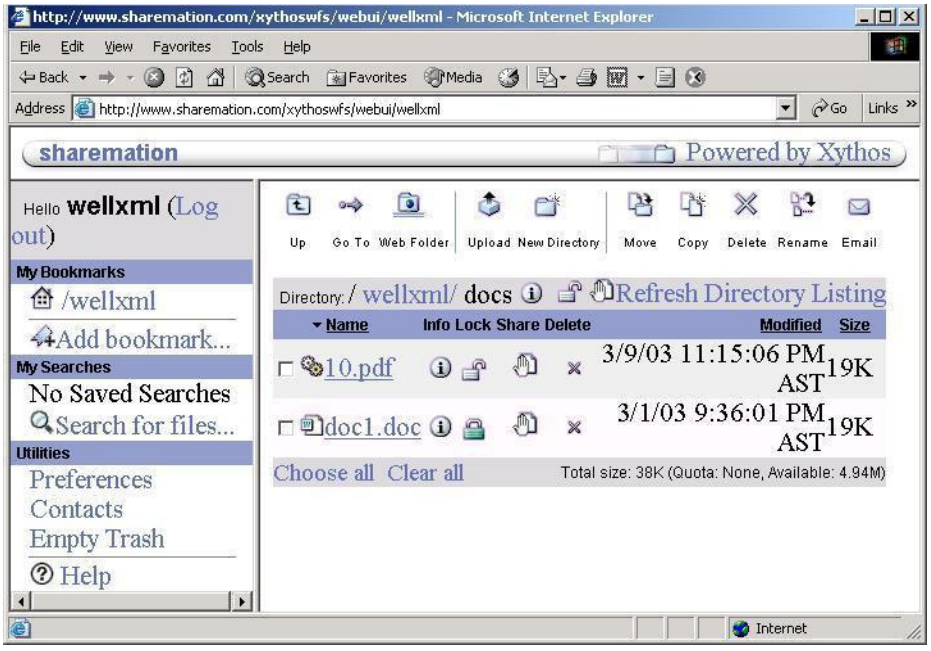


Figure 5.5: Accessing Sharemation Site by the Internet Explorer

- **Oracle9i Enterprise Edition**

Oracle9i Database Enterprise Edition provides a novel feature, called Oracle XML DB, that can also be accessed using WebDAV. Fortunately, Oracle XML DB represents a native XML database within Oracle9i. Therefore, it can be used as the XML document repository in the implemented system. There are several ways to access the XML DB such as HTTP and FTP protocols. Moreover, since it is a WebDAV repository, it can be accessed via Web Folders.

5.3.2 The Document Repository

The document repository depends fully on the WebDAV server. In other words, usually WebDAV servers determine the document storage type. For example, Apache HTTP Server utilizes the file system. In contrast, Oracle9i Database Enterprise Edition stores the document inside the database. In summary, it is possible to employ either file system of database as a document repository but according to the WebDAV server.

5.3.3 The Routing And Reminding Agent

The routing and reminding agent, developed by the researcher, represents the heart or the engine of the system. The agent communicates with WebDAV servers and manages the information of the document flows. According to this information and to the actions of the users, the agent takes the appropriate processes. Figure 5.6 illustrates the agent duties which are executed periodically. Also, it shows that the agent checks all flow progresses and according to their statuses, it takes the appropriate actions. The statuses of the flow progresses can be Null, D, W, or U.

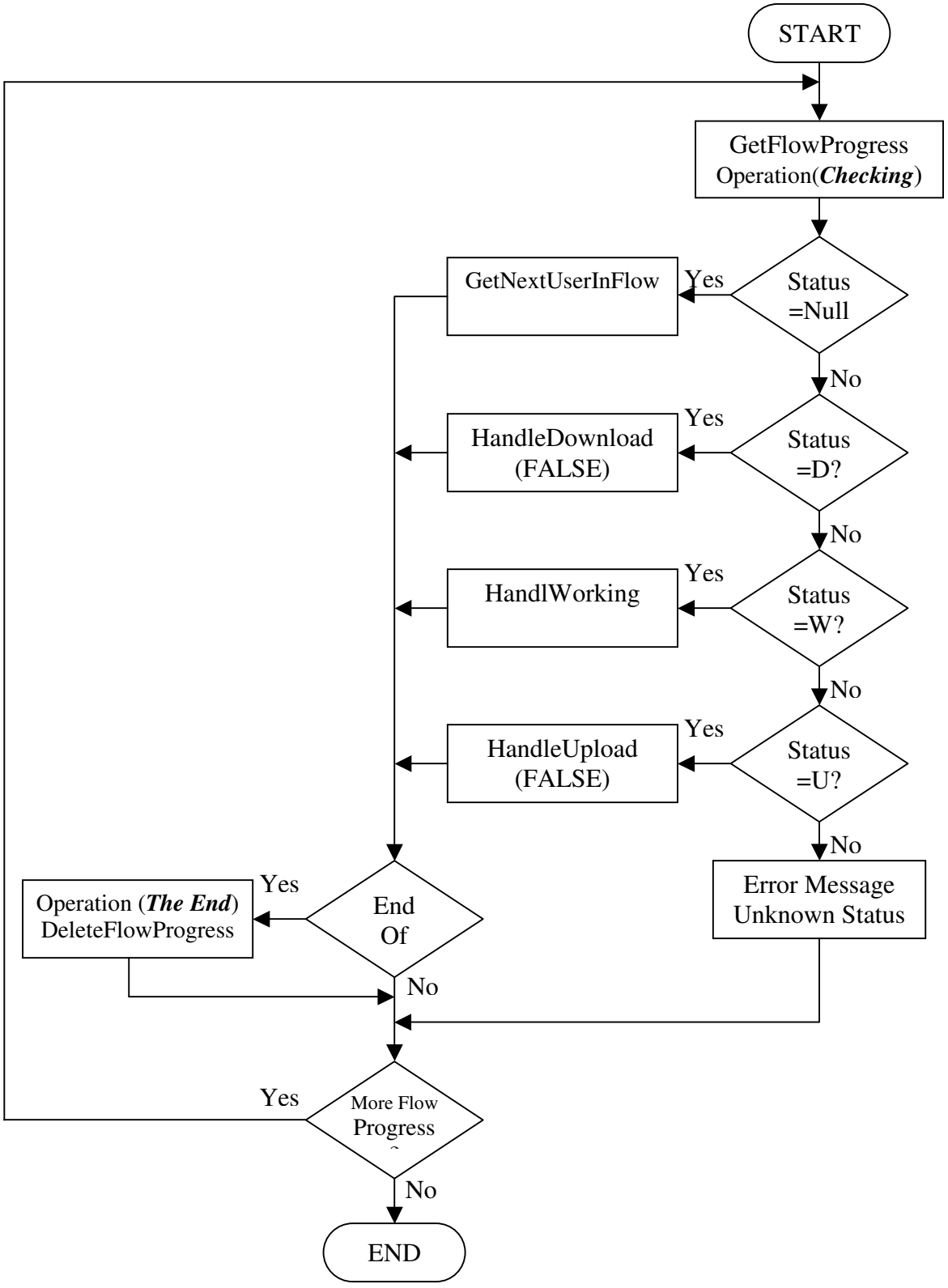


Figure 5.6: The Agent Processing Flowchart

The first status is Null which means that the corresponding flowed document starts the flow or that the current user uploads it. In either cases, the agent calls GetNextUserInFlow procedure shown in Figure 5.7. This procedure, in turn, gets the flow information of the next candidate in the flow and notifies him to download the document.

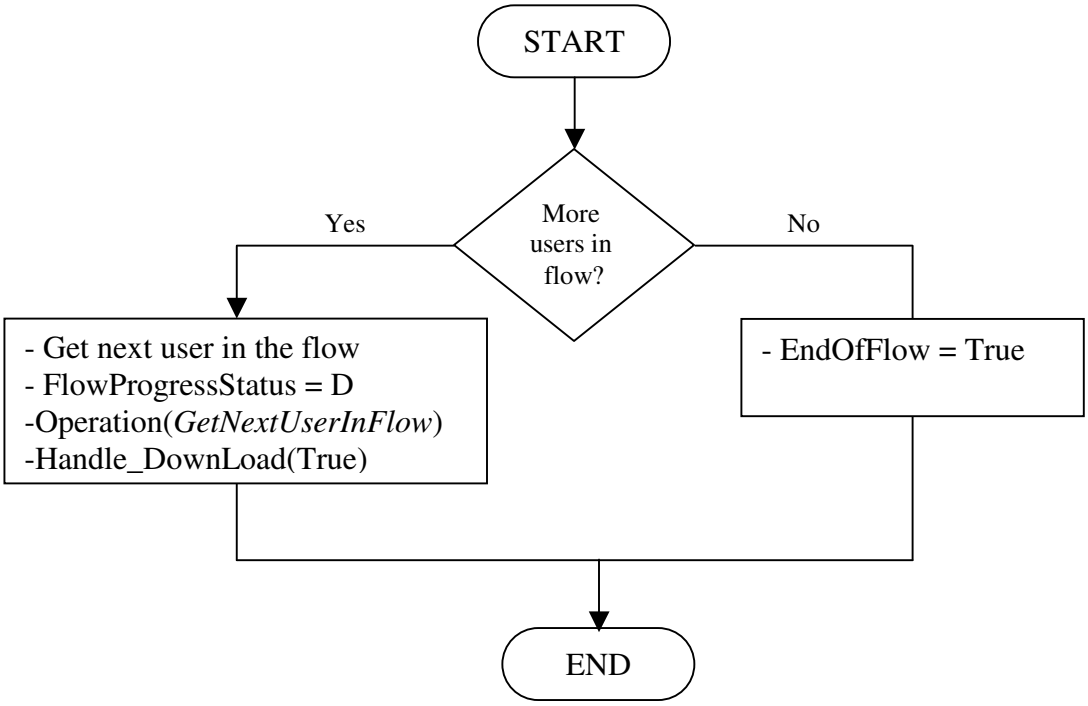


Figure 5.7: GetNextUserInFlow Flowchart

The second status is D in which the flowed document waits for the current user downloading. In this case, the agent calls HandleDownload procedure with false input variable to notify the user when necessary. The false parameter means

that this download notification is not the first one for that user. This procedure is illustrated in Figure 5.8.

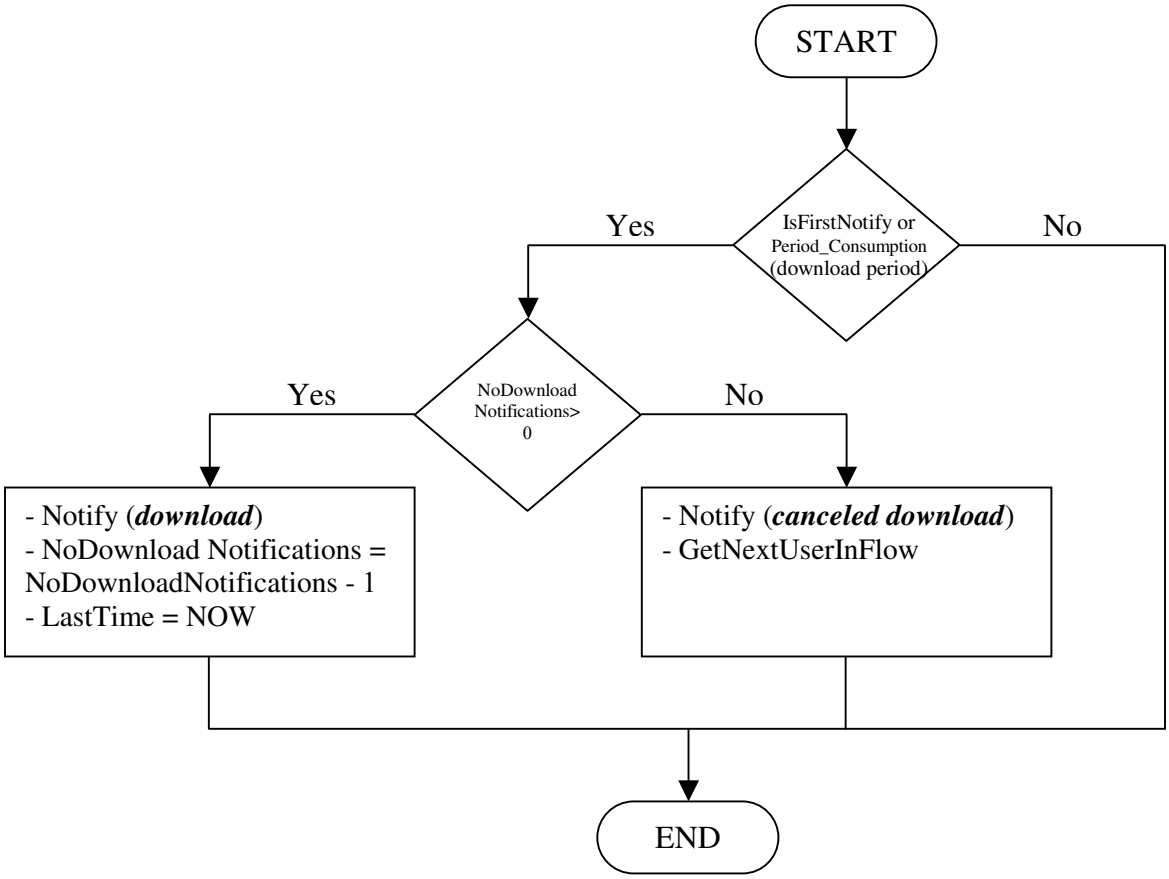


Figure 5.8 : Handle_Download Flowchart

In the same manner, the third document flow progress status (i.e. W status) denotes that the current user downloaded the document and he does not consume his working period. For this status, the agent calls HandleWorking procedure which is shown in Figure 5.9.

Finally, the forth status is U which is similar to status D except that status U is related to the uploading operations while status D is related to downloading operations. For the U status, the agent calls HandleUpload procedure which is shown by the flowchart appears in Figure 5.10.

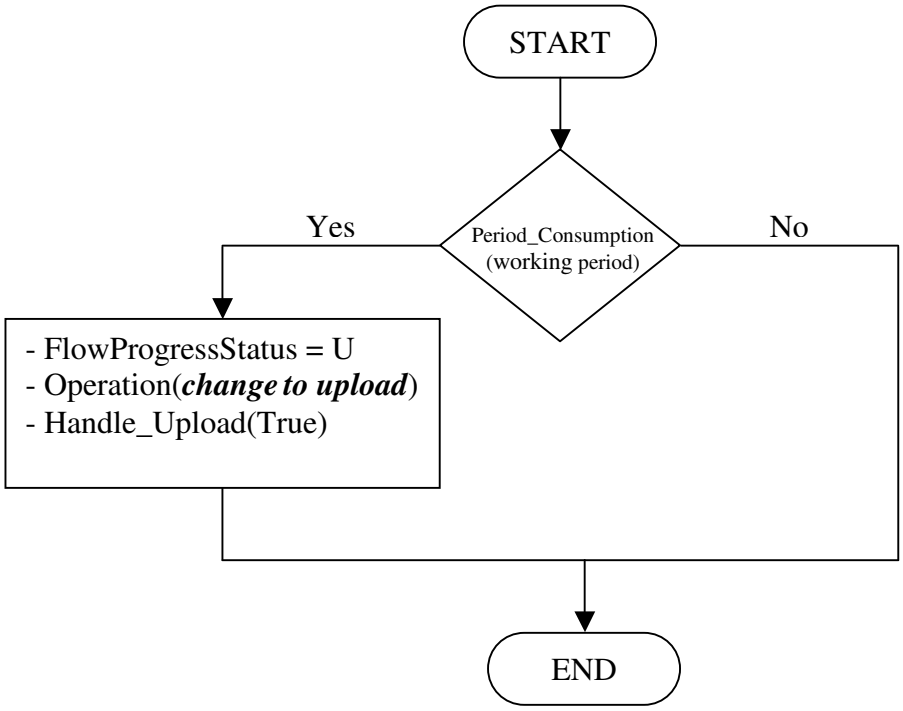


Figure 5.9: Handle_Working Flowchart

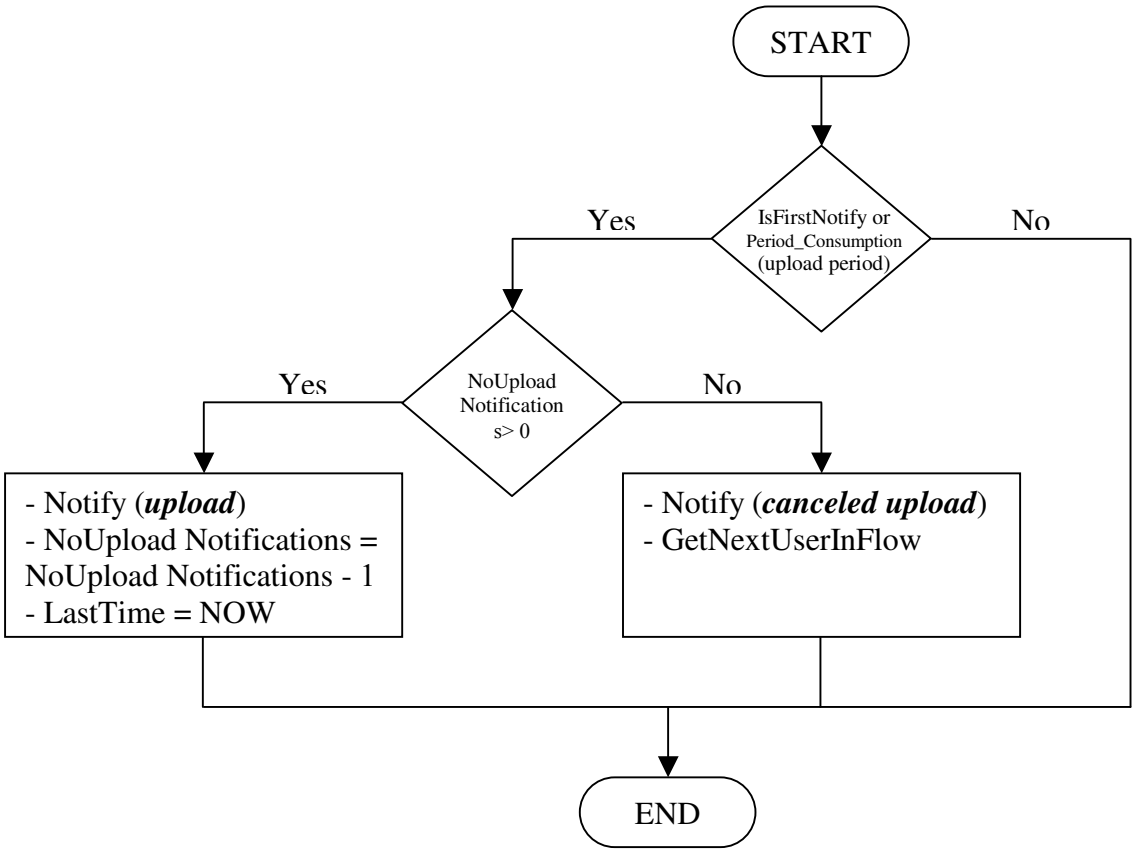


Figure 5.10: Handle_Upload Flowchart

Moreover, the HandleDownload, HandleWorking and HandleUpload procedures utilize the Period_Consumption function which returns true if the difference between the meantime and last operation is more than the specified period. The flowchart of this function appears in Figure 5.11.

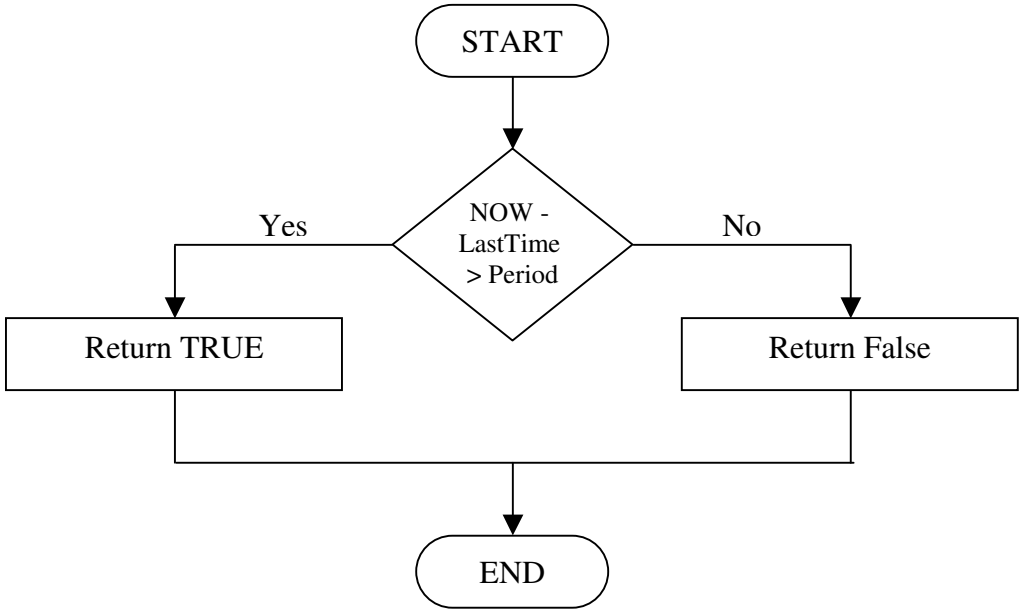


Figure 5.11: Period_Consumption Flowchart

Figure 5.12 represents the snapshot of the agent screen. It is used to trace the flow progresses.

The state diagram in Figure 5.13 shows the states of the flow progress. Initially, when a document starts a flow, it enters into NULL state. In this state the agent check if there are any more users in the flow. If there are no more users, the flow is completed. Otherwise, the agent sends a download notification to the next user and transfer the document to D state.

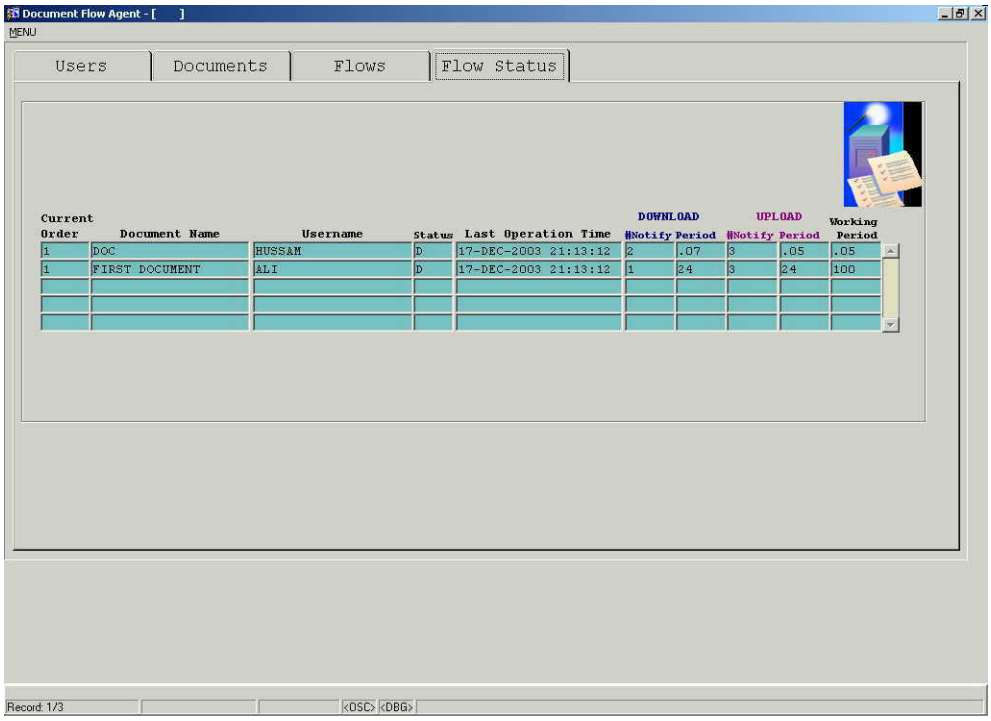


Figure 5.12: Tracing Flow Progresses Screen

State D means that the agent is waiting the user to download the document. There are three possible inputs to this state. As shown in the figure, if the user does not download the document in the appropriate time and he has more download notifications, the document stay in state D and the agent sends a notification to the user after decreasing his download notifications. The second input handles the case of the user when he does not download the document after consumption his notifications. With this input the agent notifies the user by his cancellation and transfers the document to state NULL to consider the next user. The final input represents the user when he downloads the document in which the state becomes W.

W state stands for the document after the user downloads it and before the working period is over. The state is changed either because the user uploads the document or because his working period is over. In the former case, the user finalizes his work so the document returns back to the NULL state. The latter case transfers the document to state U and sends the user the first upload notification.

The Final state is U which means that the user should complete his work and upload the document back to the server. This state is similar to D state except that U state takes care of uploading process.

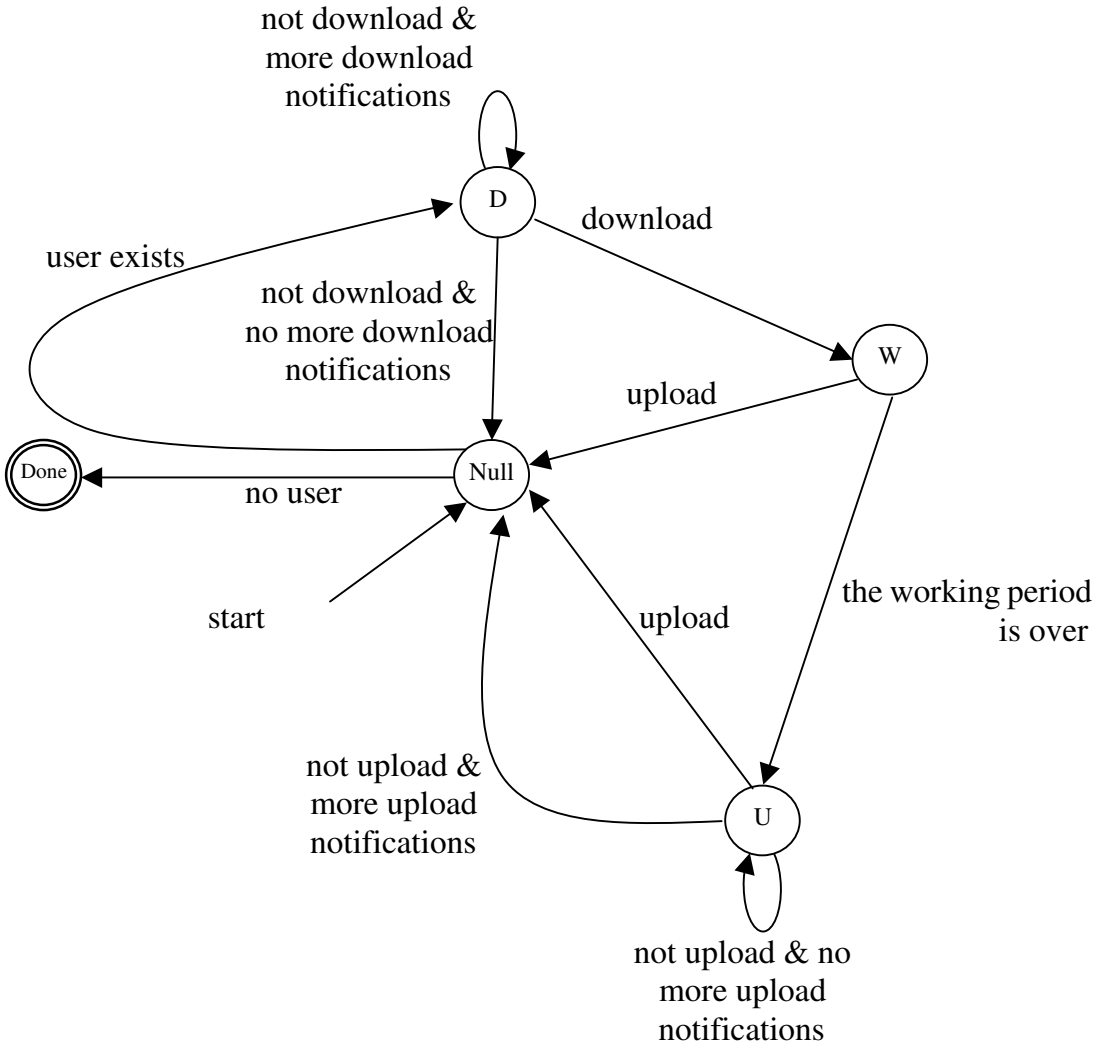


Figure 5.13 : The State Diagram of the Flow Progress

5.3.4 The Flow Repository

In this implementation, the flow information is represented and stored in a relational database. This subsection contains the description of the flow entities, relationship types and the attributes of the entities which described in Table 5.1, Table 5.2 and Table 5.3 respectively.

Entity Name	Description
User	A user is somebody who utilizes the system and associates in creating and modifying documents.
Document	A document is flowed between users according to a document flow definition.
Flow	A flow represents the definition of a possible document routing between several users.
Flow Detail	Each flow detail represents a flow definition for a user.
Flow Progress	A flow progress is updated periodically by the agent to provide the latest information about a specific document flow.
Operation	Operation are actions on documents performed either by the agent or by the users.
Notification	Notifications are sent by the agent to the users in order to encouraging them either to perform something or to provide them by suitable information.

Table 5.1 : System Entities.

Entity Name	Relationship type	Entity Name	Cardinality	Participation
User	Receives Associates Joins	Notification Operation FlowDetail	1:M 1:M 1:M	P:T P:T P:T
Document	Relates to Associates Flows using Traced by	Notification Operation Flow FlowProgress	1:M 1:M M:1 1:1	P:T P:T P:P P:T
Flow	Includes	FlowDetail	1:M	P:P
FlowProgress	Reaches	FlowDetail	M:1	P:P

Table 5.2 : System Relationship Types.

Entity Name	Attribute	Description
User	**Serial**	**Unique user identifier.**
	ID	**System logon name for user.**
	Name	**The user name.**
	Password	**The system password for user.**
	Email	**The electronic mail for user.**
Document	**Serial**	**Uniquely identifies document.**
	Name	**Name of document.**
	Path	**The repository path of document.**
Flow	**Serial**	**Unique flow identifier.**
	Name	**Flow name.**
FlowDetail	**FlowSerial**	**The serial of the related flow.**
	Order	**Placing the flow detail within the rest flow details of the same flow.**
	UserSerial	**The serial of the user in the flow detail**
	No_Download_Notify	**Number of downloading notifications.**
	Download_Period	**The period between two download notifications.**
	No_Upload_Notify	**Number of uploading notifications.**
	Upload_Period	**The period between two upload notifications.**
	Working_Period	**The period between user downloading and first uploading notification.**
FlowProgress	**DocSerial**	**The serial of the related document.**
	FlowDetailOrder	**The current flow detail order in the flow.**
	Status	**Status of flow progress which is either Null, D, W or U.**
	LastTime	**The time of the last operation or notification.**
	No_Download_Notify	**Number of remaining downloading notifications.**
	Download_Period	**The period between two download notifications.**
	No_Upload_Notify	**Number of remaining uploading notifications.**
	Upload_Period	**The period between two upload notifications.**
	Working_Period	**The period between user downloading and first uploading notification.**

Table 5.3: The Attributes of the Entities (continued overleaf).

Entity Name	Attribute	Description
Operation	**Serial** **DocSerial** **UserSerial** **Time** **Type**	**Uniquely identifies operation.** **The related document serial.** **The related user serial.** **Operation time.** **Operation type which is either Checking, GetNextUserInFlow, ChangeToUpload, EndOfFLow, UserDownloading or UserUploading.**
Notification	**Serial** **DocSerial** **UserSerial** **Time** **Type**	**Unique notification identifier.** **The related document serial.** **The related user serial.** **Notification time.** **Notification type which is either Download, CanceledDownload, Upload or CanceledUpload.**

Table 5.3 continuation: The Attributes of the Entities.

Moreover, Figure 5.14 shows the entity-relationship (ER) diagram of the implemented system.

5.3.5 The Mail Server

The mail server is responsible to receive the automatic notification messages from the routing and reminding agent via the simple mail transfer protocol (SMTP). These messages are the download and upload notifications. The download notifications ask the users to download the documents from the server and to perform their work with the documents. On the contrary, the upload notifications require the users to end their work on the documents and to upload them back to the server. Also, the mail server forwards the messages to the mail clients as will be described in the mail client subsection.

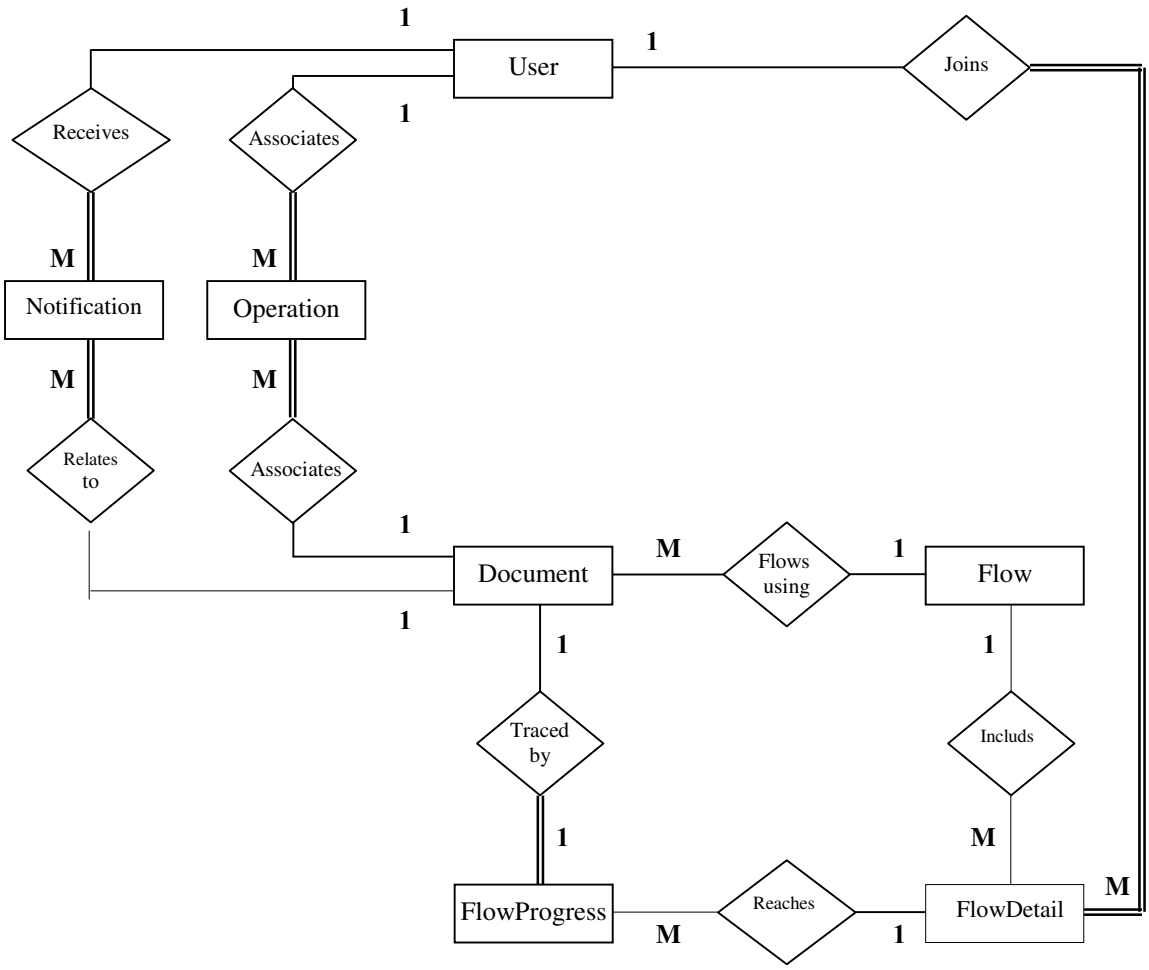


Figure 5.14 : The ER Diagram of the Document Flow Implementation

5.4 THE CLIENT SUBSYSTEM

The client subsystem includes WebDAV clients in addition to the document flow client, the local temporary document repository, non-WebDAV document editors, the flow definer and the mail client. Each of these parts will be explained in the following.

5.4.1 WebDAV clients

As shown in Chapter 4 and Appendix A, WebDAV clients can communicate with WebDAV servers to perform remote authoring of the documents through a coherent set of methods which containing request headers and request body formats. Then, servers return appropriate responses enclosing response headers in addition to response body formats. For example, WebDAV provides methods to store and retrieve resources, to create and list contents of resource collections, to lock resources for concurrent access in a coordinated manner, and to set and retrieve resource properties. Thus, WebDAV clients are the main part of the client subsystem of the system. In the implementation, I used several WebDAV clients including DAV Explorer, Tamino WebDAV Basic Client, Web Folders, Microsoft Office, Internet Explore, jEdit and XML Spy. These enormous products of WebDAV clients make it possible to upload and access documents stored in WebDAV repositories using standard, familiar interfaces as will be shown.

The WebDAV clients differ in their behavior and according to that can be divided into four groups: dedicated WebDAV clients, WebDAV-enabled clients, WebDAV document editors and web browsers. In the following, several WebDAV clients are elaborated for each group.

The first group represents the dedicated WebDAV clients which includes DAV Explorer and Tamino WebDAV Basic Client. These clients send WebDAV requests directly to the server. The DAV Explorer is similar to the Windows

Explorer in its interface as shown in Figure 5.15. It translates the user actions to the proper requests which are sent to the server.

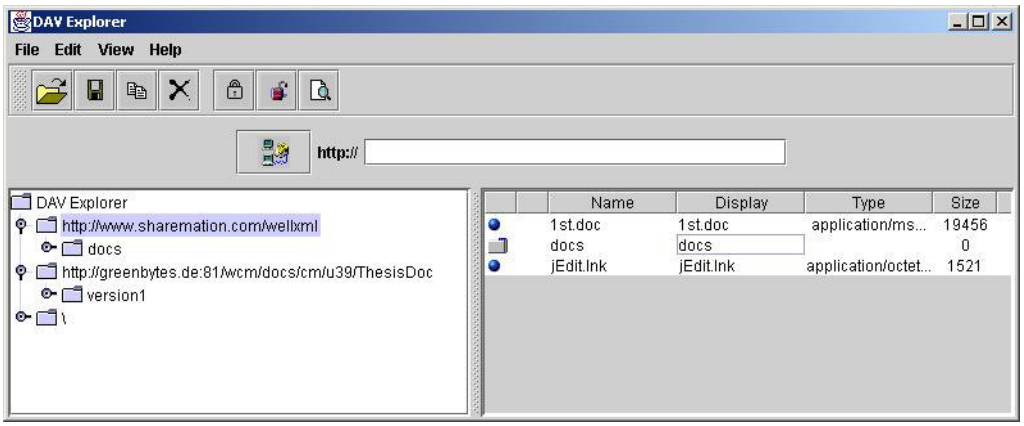


Figure 5.15: The DAV Explorer Interface

The Tamino WebDAV Basic Client allows its user to write the requests, send them and check their responses from the server. The advantage of this client is that it enables users to send any request type directly, including versioning requests. Figure 5.16 presents the interface of this client.

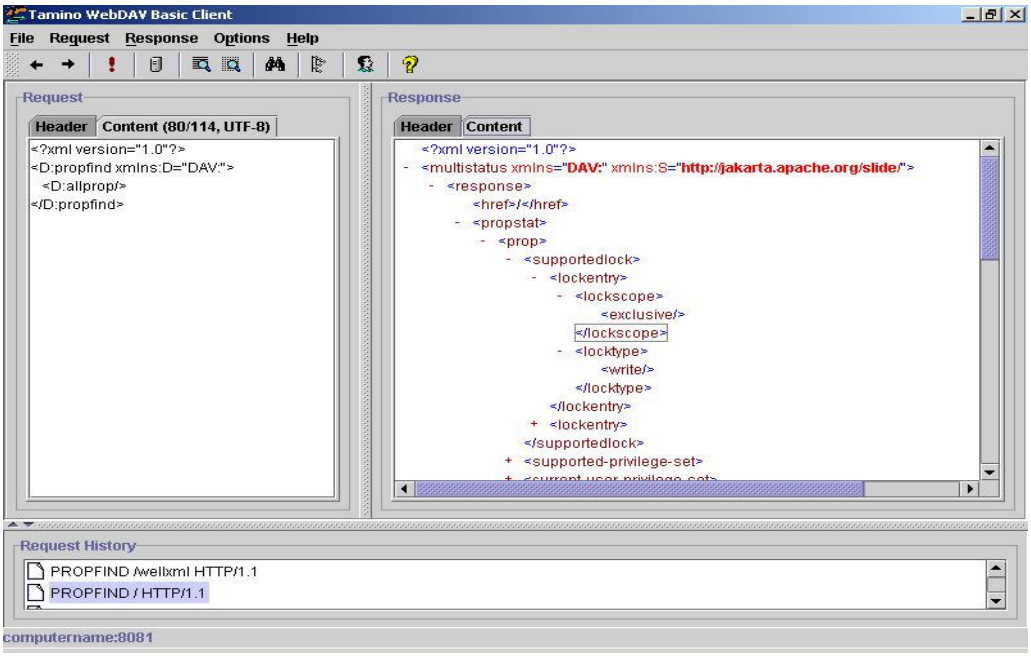


Figure 5.16: Tamino WebDAV Basic Client

The second group represents the WebDAV-enabled clients, supporting the WebDAV protocol via Web Folders. This group represents the most popular way of accessing WebDAV servers on Microsoft Windows. A Web Folder represents an interface to a WebDAV repository on the WebDAV server and so Web Folders can act as an intermediate medium between WebDAV-enabled clients and WebDAV servers. Figure 5.17 shows several definitions of WebDAV repositories used in the implementation.

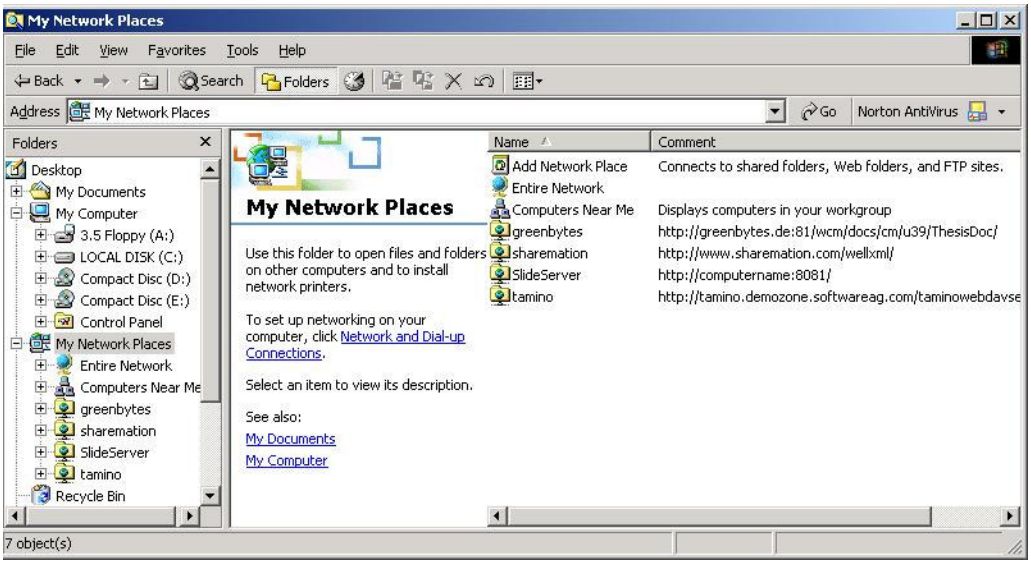


Figure 5.17: Accessing WebDAV Repositories via Web Folders

Moreover, Microsoft Office is a WebDAV-enabled client which allows direct editing of the documents in the WebDAV server repositories via Web Folders as appears in Figure 5.18. Another interesting feature of Web Folders is the simple deal with the WebDAV repository. It is possible to drag and drop files into and from Web Folders since they can be handled as normal folders as shown in Figure 5.19.

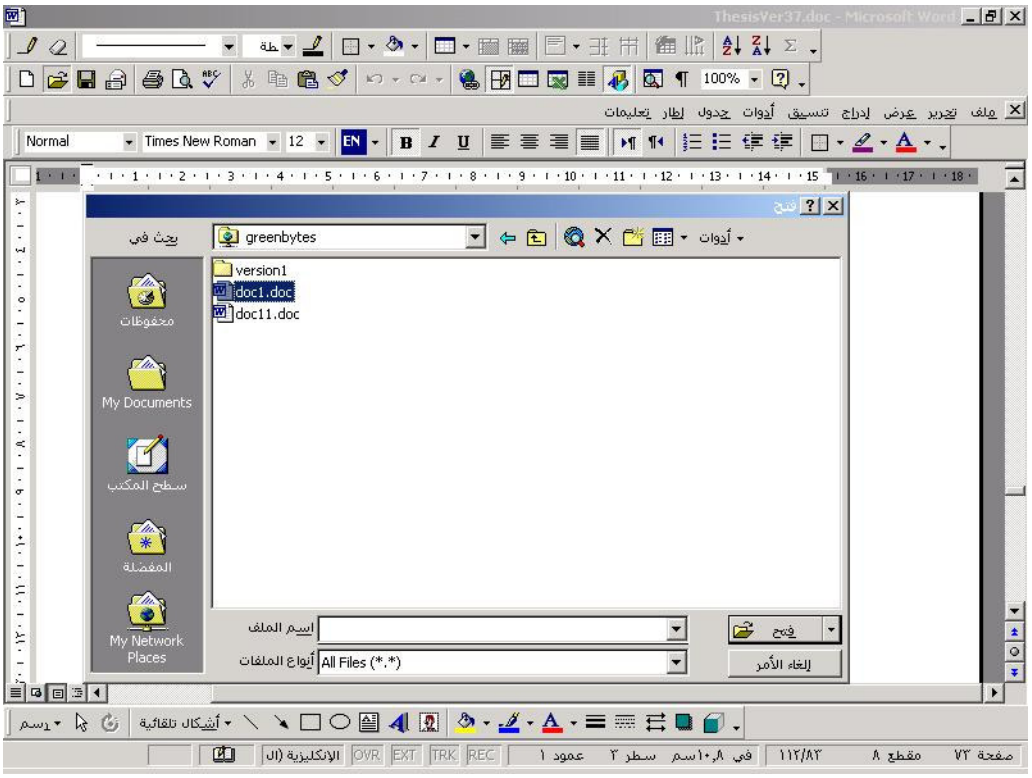


Figure 5.18: Direct Editing of WebDAV Documents using Microsoft Office

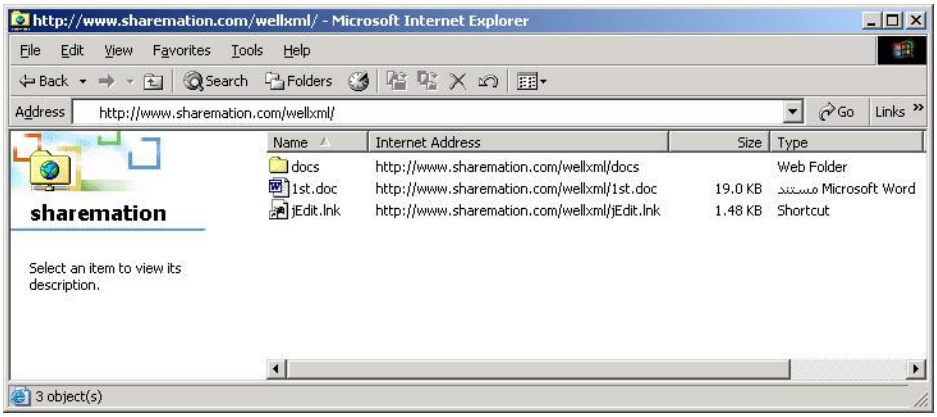


Figure 5.19: Accessing WebDAV Repository via a Web Folder

The third group of the WebDAV clients includes WebDAV clients which can be used to edit documents directly on WebDAV servers without Web Folders. For example, JEdit, which is Java-based open source text editor, can be extended to be a WebDAV client using a WebDAV plug-in as shown in Figure 5.20. Then by using locks to prevent lost updates, it can be used to edit text files directly on WebDAV servers.

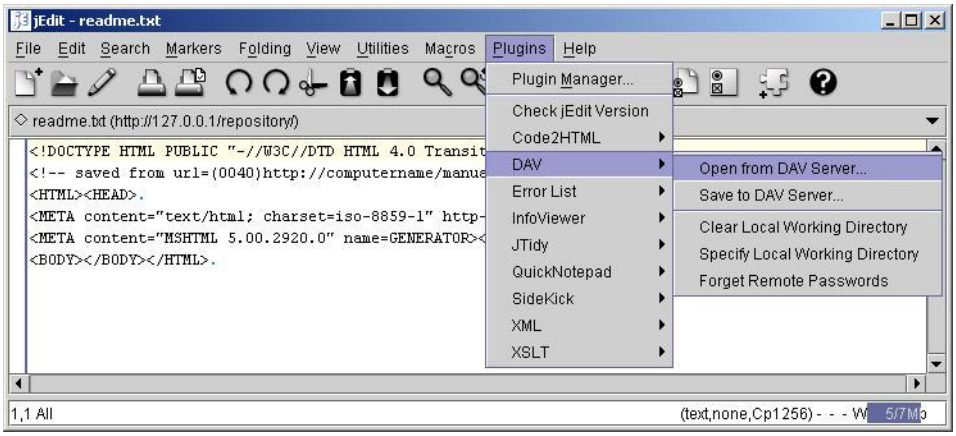


Figure 5.20: The JEdit with WebDAV plug-in

Also, XML Spy, which is an XML development environment, can communicate directly with several XML WebDAV repositories. Moreover, it supports opening and saving documents from and to WebDAV servers as shown in Figure 5.21.

The fourth group of the WebDAV clients is the web browsers such as Internet Explorer. But the web browsers may face some limitations when dealing with WebDAV servers because the current web browsers do not support WebDAV methods via HTTP protocol. In this case, the browsers can browse the documents on the repository and download them only as previously shown in Figure 5.3 of Tamino WebDAV Server. In contrast, some WebDAV servers provide web applications which allow the web browsers to utilize the WebDAV repositories in a smart way

and friendly interface. For example, SAP Portals Enterprise Portal Server and Xythos WebFile Server can be utilized via web browsers as shown in Figure 5.4 and Figure 5.5.

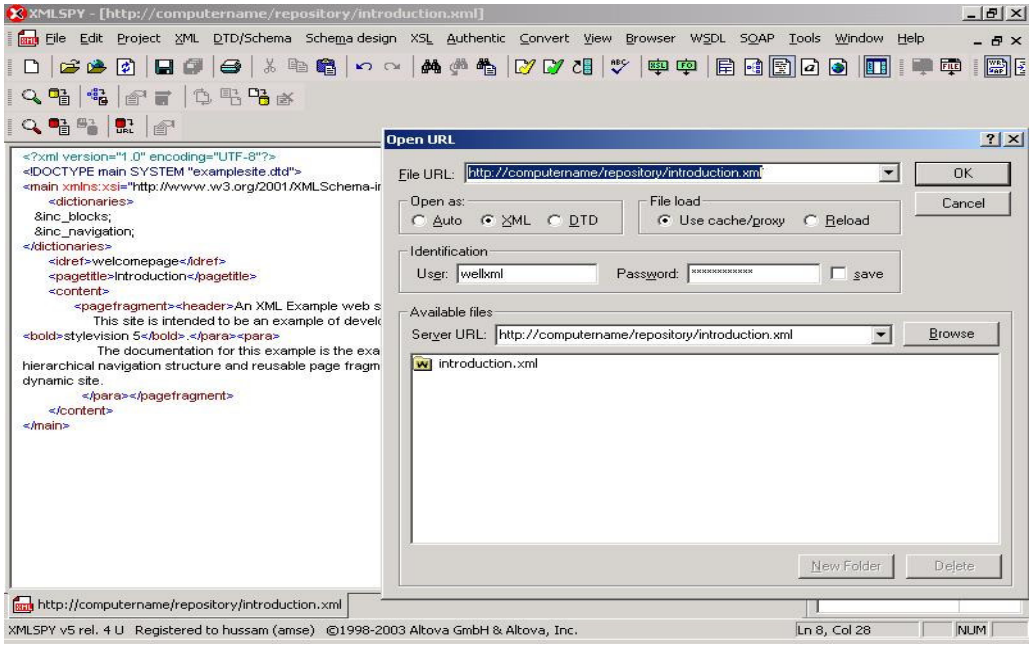


Figure 5.21: XML Spy Accessing WebDAV Repository

5.4.2 The Document Flow Client

I implement a special client which helps users to identify and handle their work. Figure 5.22 shows the first screen of the document flow client in which users download and upload documents according to their privileges. Also, Figure 5.23 shows the second screen of the client which inform each user with notifications sent to him by the agent.

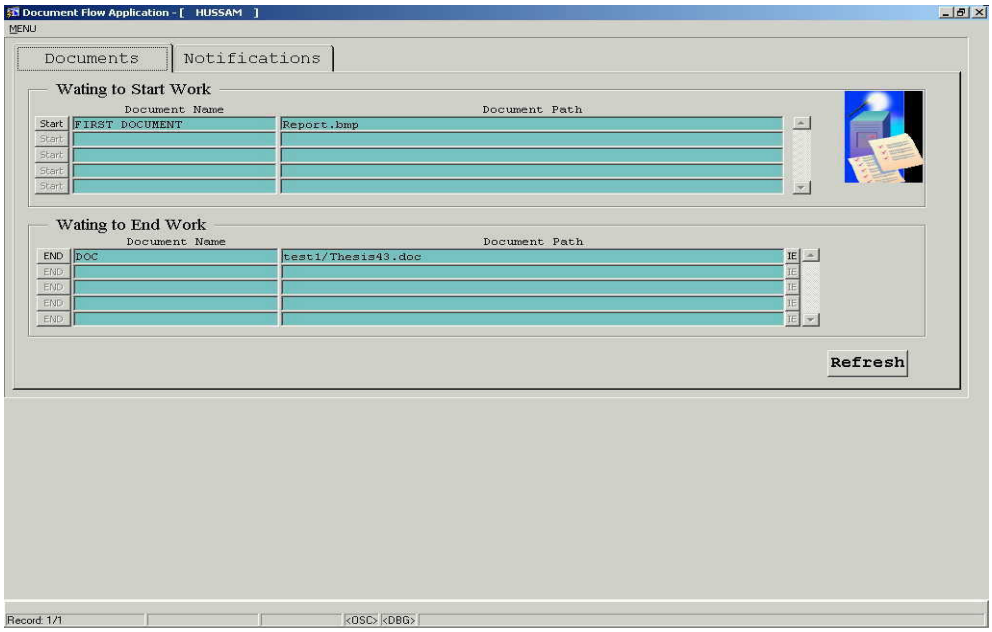


Figure 5.22: The Documents Screen of the Flow Client

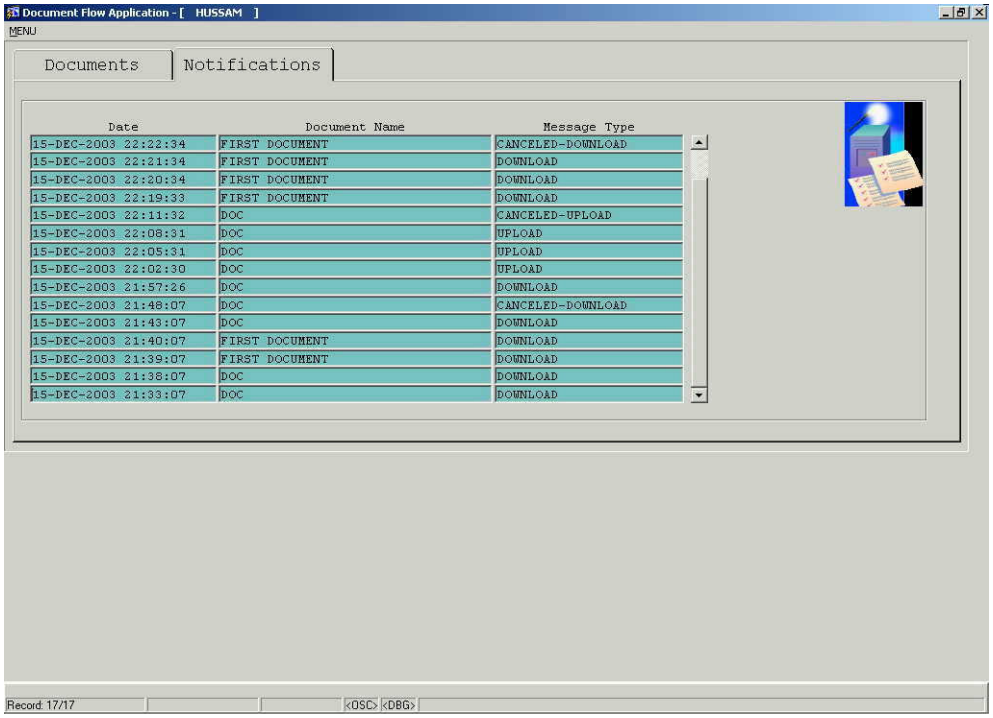


Figure 5.23: The Notifications Screen in the Flow Client

Figure 5.24 present the flowchart of the user downloading processes. According to this figure, when a user downloads a document, its flow progress status becomes W which makes the agent waits for the document working period before notifying the user to upload the document. Similarly, Figure 5.25 shows the flowchart of the uploading processes, in which the flow progress status is set Null to provide an indication for the agent to deal with the next user in the flow.

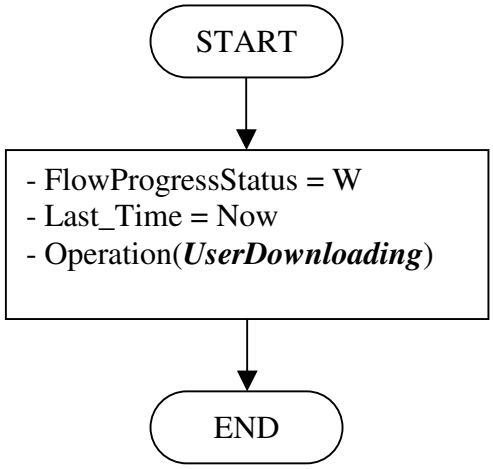


Figure 5.24: User Downloading Document Flowchart

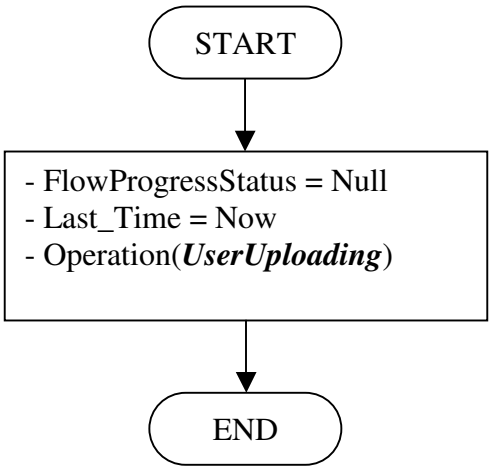


Figure 5.25 : User Uploading Document Flowchart

5.4.3 The Local Temporary Document Repository

The local temporary document repository is an optional component on the client subsystem. It can be used either because the editors do not support WebDAV protocol or because the communication media is not reliable. In both cases, users can download the documents from the WebDAV server to their local temporary document repository where than can edited the document. After finalizing the editing, they upload the documents back to the server.

5.4.4 Non-WebDAV Document Editors

Any document editor can be used in the implemented authoring system. But document editors differ in the mode of editing documents in WebDAV repositories. As shown in the WebDAV clients' subsection, several document editors can edit documents directly in WebDAV repositories without downloading them. Moreover, any editor can be used to edit documents locally after downloading them from the servers.

Several document editors can communicate with WebDAV repositories either directly or via Web Folders as shown in the previous section. In both cases, it is possible to edit documents in WebDAV repositories directly by using the editors. Thus, wide applications can be used as document editors in the implemented system including: Microsoft Office, JEdit and XML Spy.

Moreover, the system can get benefits from non-WebDAV editors by editing documents locally in the client subsystem after downloading them from servers. To prevent the lost update problem, documents are locked before they are downloaded. After editing them locally, they can be uploaded back to the server where they are unlocked. In summary, any document editor can be utilized in the authoring system by employing a lock-download-work-upload-unlock authoring process.

5.4.5 The Flow Definer

The document flow definer is responsible to define document flows. As previously shown in Chapter 3, the flow of a document includes the users who are responsible to work with the document. Also, the flow defines for each user five attributes: the number of downloading notifications, the downloading notification period, the number of uploading notifications, the uploading notification period and the working period. The number of downloading notifications and the number of uploading notifications of a specific user are the number of the notifications which the agent should send to the user remind him to download or to upload the document. The download notification period and the upload notification period represent the period between two download or upload notifications. The working period is the period between downloading the document and sending the first uploading notification. The roles of these attributes are clarified early in Figure 3.3. In the implementation, the flow information is saved in a database system in the server subsystem as stated in subsection 5.3.4. Some snapshots of the flow definer

screens appear in Figure 5.26, Figure 5.27 and Figure 5.28. These screens are used to define users, documents and document flows respectively.

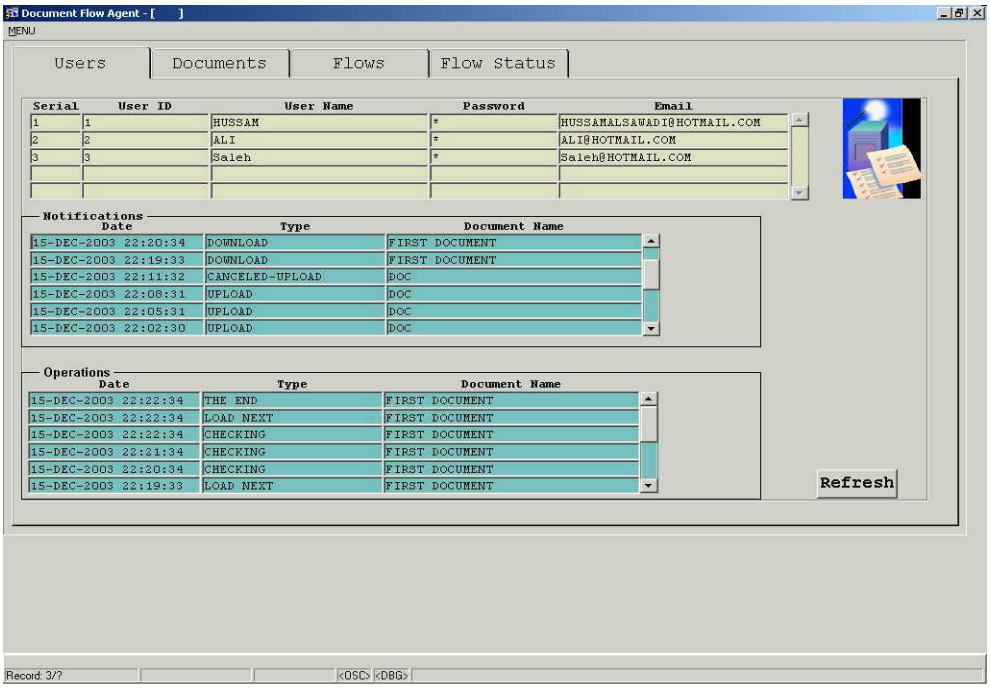


Figure 5.26: Defining Users Screen

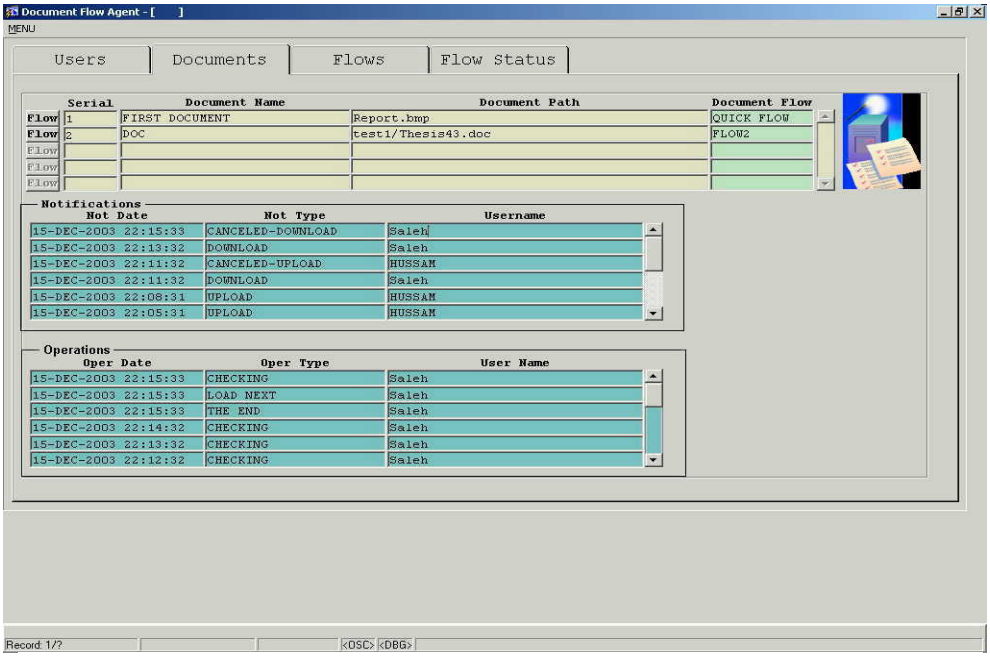


Figure 5.27: Defining Documents Screen

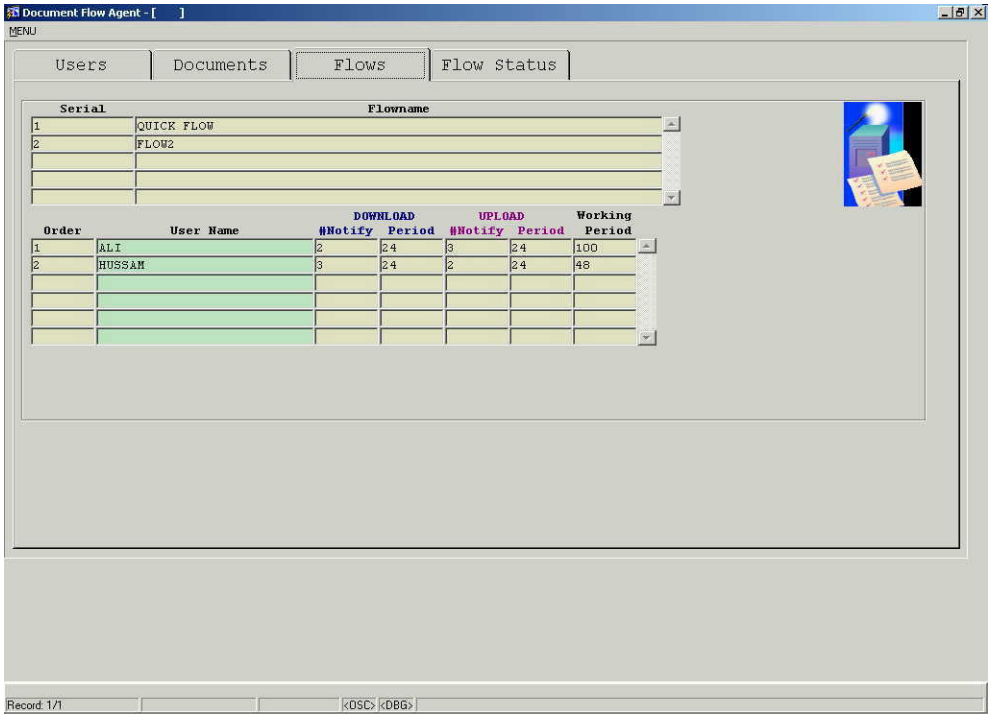


Figure 5.28: Defining Flows Screen

5.4.6 The Mail Client

The mail client in the client subsystem is used in the system to receive the agent notifications in the picture of emails. The email notifications provide users with appropriate information including the actions which they should do as explained previously in Chapter 3. The email notifications include download, upload and cancel notifications. A download notification notifies a user to edit a document when it is his turn in the flow path or when he does not do so in time. This notification informs him about his turn in the document flow and requests him to download the document to perform his work with it. In contrast, an upload

notification prompts a late user to finish his work with a document and upload it back to the server. It will be sent to him if he does not upload the document after the working period is over. Furthermore, a cancel notification informs a lazy user of canceling him from the flow when he does not do his work by downloading or uploading a document in the suitable time. Figure 5.29 shows an automatic generated email notifying a user to download a document.

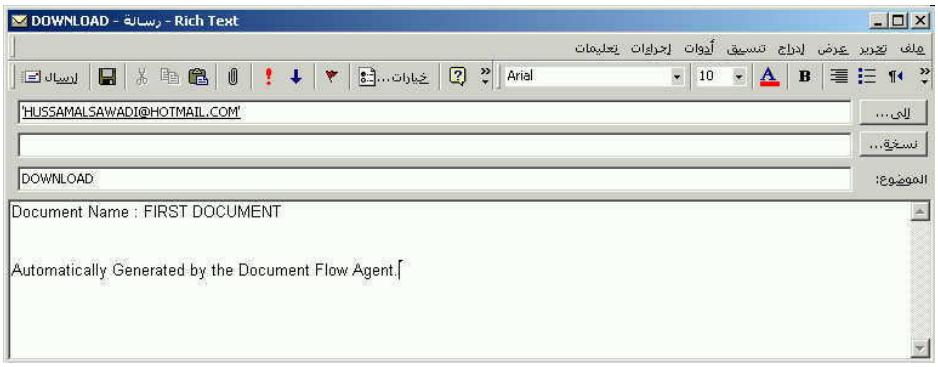


Figure 5.29: An Email Generated by the Agent

CHAPTER 6

COMPARISON AND ANALYSIS

This chapter compares the literature survey systems according to a set of comparison criteria. Next an analysis of the comparison is provided. Finally, it presents an evaluation of the DFWDAV model using the same comparison criteria. The objective of this comparison is to show the improvements which the DFWDAV model provides.

6.1 COMPARISON

Each of the literature survey applications contains a document flow and archival model. These document flow and archival models vary in their architectures and techniques. Several factors contribute to this variation. The most important factors are the objectives of their applications and the technologies available at the time when they were developed. This section briefly points out the differences between these models.

The models are briefly compared from several phases in Table 6.1. This comparison is divided into two parts: first, document flow, and second, document archival.

Model / Factor	EDMS	Alliance	DocMan	DReSS
Editing	Lock, download and edit locally	By holding a master copy and edit it locally	Lock, download and edit locally	Lock, download and edit locally
Flow mechanism	Between the server and the clients Using state graph	Fragment migration between sites	Between the distributed store and local disks.	Between the server and the clients
Document syntax	Various engineering Documents	Special format	Miscellaneous	Any documents
System platform	Unix & X Windows	Unix	Server: Solaris Client Windows	Unix or Windows NT
Network	LAN	WAN	WAN	The Internet
Dependency	Database & special software	Special software	Special software	Special software
Document repository	Relational database	File system	File system	Database & file system
Document location	Centralized	Distributed & replicated between sites	Distributed between the sites	Centralized
Automatic Notification	Internal asynchronous notification between the server and the clients	Internal notification to update the fragments	Two type: - Automatically reflect the changes of the document folders. - Subscribing to email notification.	Not available
Versioning	Available	Not available	By creating a copy of the updated document after finishing the updating	Not available
Access Control	On operation level using privileges	On fragment level using the roles	By defining user access	Available
Handling Collaborative functions	In the application	In the application	In the application	In the application

Table 6.1: Comparison between Different Models (continued overleaf).

Factor \ Model	Oxford Radcliffe Hospital document management system	BSCW	Smartcard system	EnAct
Editing	Not available	Lock, download and edit locally	Not Available	By MS Word directly to the repository
Flow mechanism	Between authors and readers	Between the server and the clients	From a sender to a receiver	By Assigning a document to the users
Document syntax	SGML	Any document	XML	SGML
System platform	Not mentioned	Unix Or NT	Any system	Unix Or NT with PCs
Network	Intranet	The Internet	The Internet	The Internet
Dependency	Database	Extension to a Web server and database	Smartcard & Smartcard reader	SIM (Structured Information Manager) and customized MS Word
Document repository	Database	Database & file system	Not Available	SGML Repository
Document location	Centralized	Centralized	Distributed between the sites	Centralized
Automatic Notification	Using author distribution lists and reader subscriptions	By event service	Not Available	Not Mentioned
Versioning	Not available	Linear versioning	Not Available	Linear versioning
Access Control	Not available	On workspace level	Only the receiver can access the document	On the document level
Handling Collaborative functions	In the application	In the application	In the application	In the application

Table 6.1 continuation: Comparison between Different Models
(continued overleaf).

Model / Factor	An extension of Lotus Notes	AllianceWeb	GroupWriter	WebDAV system
Editing	Any time by creating new revisions before the editing	By holding a master copy and edit it locally	Using Check out and check in	Lock, download and edit locally
Flow mechanism	Depending on Lotus Notes	Fragment migration between sites	Direct updating or by annotations	Between server and clients
Document syntax	HTML	HTML and XML	TXT or RTF	Any document
System Platform	Any one supports Lotus Notes	Any system	Windows	Any platform
Network	Intranet	The Internet	LAN and dial up	The Internet
Dependency	Lotus Notes	Extended WebDAV server and Amaya	Special software tools	WebDAV server (Apache and mod_dav) and an extension to a Web browser
Document repository	Lotus Notes repository	File system	File system	File system
Document location	Centralized	Distributed & replicated between the servers	Centralized	Centralized
Automatic Notification	Not Mentioned	Internal notification to update the fragments	Not Mentioned	Not available
Versioning	After every modification	Not Available	Available temporarily	Not available
Access Control	On the document level	On document fragment level	On documents level	On document level
Handling Collaborative functions	In the application	In the application	In the application	At Network Layer

Table 6.1 continuation: Comparison between Different Models (continued overleaf).

Factor \ Model	MS Office 2000 Annotation System	Form Flow Model	X-Folders
Editing	Each user can update his annotations	By updating database field	Directly on the repository
Flow mechanism	Adding and reading annotations	Not Mentioned	Either by transferring the documents between the folders or by chaining the status of documents
Document syntax	HTML	Forms only	Any document
System Platform	Windows	Any platform	Any platform
Network	Intranet	Based upon TCP/IP	Internet
Dependency	MS Office 2000, WebDAV Server, SQL Server and annotation client	JATLite (An agent platform) and KQML	WebDAV servers
Document repository	Database for annotations and file system for documents	Database	File system
Document location	Centralized	Centralized	Centralized
Automatic Notification	By subscription	Not mentioned	Not mentioned
Versioning	Not available	Not available	Not available
Access Control	Each user can update only his annotations but can read all annotations	Depends on database privileges	Depends on database privileges
Handling Collaborative functions	In the application	In the application	At Network Layer

Table 6.1 continuation: Comparison between Different Models.

The first part of the comparison includes three tasks: document editing, flow mechanism and automatic notification. Some models do not allow the first task (i.e. document editing) because their applications do not require this task. The other models accept different editing methods. Most models, for example, apply a locking mechanism which allows the user who locks and downloads a document to edit this document. The lock will be held for him until he uploads the document. In contrast, the extension of Lotus Notes and MS Office2000 annotation models apply different kinds of methodologies. The extension of Lotus Notes mode allows any user at any time to update a document by initiating and saving a new version of it after the modifications have been accomplished. The MS Office 2000 annotation model does not allow its users to modify any annotation except for its initiator.

The next task is the flow mechanisms in which models imply wide variations. Some models do not have a structured manner to handle the document flow because the actions of the users determine the document flow. Thus, the privileged user who firstly locks a document is the only one who can download and update it in his local machine. In contrast, EDMS innovates the document flow state graph. Also, Alliance and AllianceWeb apply the document fragments flow automatically. Similarly, the Smartcard system determines the receiver of each document, and the EnAct system assigns each document to the users.

The final document flow task is the automatic notification which enhances the flow since the users can take the right decision based on the notifications. While most models do not provide any notification mechanism, some of them (e.g. EDMS and Alliance) have an internal notification between the model components. This

internal notification updates the components information. In addition, some models provide high-level notification. For example, the Oxford system notifies the users of any new concerned document because they subscribe in its subject or because its author adds them in the document distribution list. Also, BSCW offers a thorough event service which allows any member in any workspace to define several documents or workspace events. Finally, Office 2000 annotation system allows its users to subscribe in a document to receive information about its annotations.

The second part of the model comparison (i.e. document archival) consists of the document repository and document format. Most models rely on a centralized repository, while other models have either a distributed or a replicated or no repository. DocMan, for instance, is based upon a distributed repository, whereas Alliance is based upon a replicated repository, and finally the Smartcard system does not have a repository at all. In addition to the above mentioned, the repository can be divided into several groups based on the storage method: database repositories, file system repositories and mixed database and file system repositories. Table 2.1 shows the repository type of each model. Finally, some models rely on a ready repository such as EnAct that uses a SGML repository and the extension of Lotus Notes model which depends on the Lotus Notes repository.

The other factor which contributes to comparing document archival part is the document format. Several models do not enforce the users to use a specific document format such as EDMS, DReSS, BSCW and WebDAV system. In contrast, some models work with their own special document format, like Alliance. The rest

are based upon a known document format such as text format or markup languages (i.e. SGML, HTML and XML).

6.2 ANALYSIS

Several reasons lead to the inconsistency between the literature survey document flow and archival models. The most influential reasons are the objectives of their applications and the available technologies when they were developed.

The first factor which has the major effect is the impact of the application objectives. These objectives include cooperative authoring, document managing, Internet document publishing, interested document acquiring, secure document sending and document annotation exchanging. The distinctive objectives yield distinctive functionality. The models, for example, which support cooperative authoring handle the document updating issues, and most of these models provide versioning faculties. In contrast, the Oxford system is designed to inform its users about the availability of novel interest documents. Therefore, this model does not bother itself with the updating or versioning functions. Also, the Smartcard system does not handle these two issues because its objective is just the secure document exchange over the Internet.

The second influential factor on the document flow and archival models is the effect of the technologies used in their applications. It is clear from Table 6.1 which is ordered by the development year of the models that most models are affected by the technologies available during the period of their development. For

example, all models designed after 1997 support the Internet environment except GroupWriter because its objective is justified without the Internet. Moreover, most of these models utilize at least one of the recent markup languages (i.e. HTML and XML). Also, the recent models support the Windows operating system while the models before 1995 work only on the Unix environment. Another example, the system extended Lotus Notes and the Office 2000 annotation system rely on well-known commercial products (i.e. Lotus Notes and Office 2000). Finally, several recent systems, specifically AllianceWeb, WebDAV system, Office 2000 annotation, utilize the novel protocol WebDAV that transfers the complexity of handling the collaborative issues from the application level to the communication protocol level.

6.3 COMPARISON BETWEEN THE DFWDAV MODEL AND OTHER MODELS

The implementation shows that the DFWDAV model overcomes some of the limitations of the literature survey models which are investigated in Chapter 2. Moreover, the implementation demonstrates several of the robust features of the DFWDAV model. The objective of this section is to compare the literature survey models with the DFWDAV model. This comparison will illuminate some of the DFWDAV model characteristics.

Since the literature survey models are compared previously in Table 6.1 according to several different factors, the DFWDAV model is compared with other

models using these factors. Table 6.2 summarizes the DFWDAV model characteristics according to the comparison factors.

Model / **Factor**	**DFWDAV**
Editing	Either direct editing of documents in the repositories or by downloading documents locally where they can be edited.
Flow mechanism	The flow of the documents between users is a structured and automated flow but according to the document flow definitions.
Document syntax	Any document syntax.
System platform	Any platform.
Network	The Internet.
Dependency	WebDAV server and clients.
Document repository	Can be a database repository or a file system repository depending on the WebDAV server.
Document location	Centralized
Automatic notification	Available to the users in the flow definitions to remind them to download or upload documents.
Versioning	Available via WebDAV protocol.
Access Control	On the document level.
Handling Collaborative functions	At Network Layer (i.e. via WebDAV protocol)

Table 6.2: The DFWDAV Model Characteristics According to the Comparison Factors

In addition, the comparison will depend on the comparison parts presented in section 6.1 which are divided into two parts: document flow and document archival.

The document flow part includes three factors: document editing, flow mechanism and automatic notification. According to the first factor, the DFWDAV model is very flexible since it allows two different document editing methods depending on the document editors: direct document editing and lock-download-work-upload-unlock editing. As shown several document editors can edit documents directly in the repository if they are WebDAV clients such as XML Spy and Microsoft Office. Moreover, if they do not support WebDAV protocol or the commutation media is not reliable, it is possible to lock and download documents from the repository to the clients where they are edited. Then, they can be uploaded back to the repository where they are unlocked. The second factor of the document flow comparison is the flow mechanism in which the DFWDAV model handles the flow of documents in a structured manner. The model relies on document flow definitions to determine the flow of documents. At any time only one user is allowed to edit a specified document. On other side, documents may flow physically between server and clients if they are edited locally. The third and final document flow factor is the automatic notification. Clearly, the DFWDAV model provides high level user notifications which enhance the flow of documents and increase the speed of producing documents. As shown, there are three notification types: downloading, uploading and canceling notifications.

On the other hand of the comparison, the document archival part consists of two factors: the document repository and the document format. With respect to the former factor, the document repository of the DFWDAV model is centralized. Furthermore, it depends completely on the WebDAV server. Although several WebDAV servers provide file system repositories, other servers provide database repositories. This variety offers more flexibility in the implementation wise. The latter factor (i.e. document format) shows the reliability of the DFWDAV model. As shown, the model does not enforce a specific kind of document formats as it supports numerous textual documents. Moreover, it also applicable to non-textual documents such as drawing documents.

In summary, the DFWDAV model promises flexible editing methods and distinctive document repositories. It is independent of document syntax and applied software products (i.e. editors, clients and servers). Moreover, it ensures structure document flows, high level notifications and document versions. Most of these features result from the full utilization of the WebDAV protocol. This utilization transfers the complexity of handling collaborative issues from the application side to the network layer, and so the DFWDAV model concentrates on the flow issues such as flow defining, analyzing and processing.

CHAPTER 7

CONCLUSION

Most of the available document collaborative applications lack a systematic document flow module. In these applications, the document flow functions are incorporated within other functions. Therefore, it is very important to design a document flow model which can be used as an infrastructure for such applications.

In this research, I have developed DFWDAV (a document flow model) that can support modern web cooperative applications. I believe that the primary value of this work comes from the modularity and the systematically of DFWDAV whereas most other applications combine this functionality with other functions.

Despite the fact that this research has a strong structure flavor, the work presented in this thesis strongly surpasses most of the applications that handle documents in collaborative environments. While these applications provide wide functionality, they do not consider the document flow as a standalone function. In contrast, this research has provided a generic and systemic document flow model which utilizes innovative technologies. Through the examination of existing systems, the model reported in this thesis has sought to reflect a generic infrastructure for collaborative applications. The reflection is used to design a generic systematic document flow model.

The rest of this chapter is organized as follows. Section 7.1 summarizes the main achievements of the research presented in this thesis. Section 7.2 identifies relevant issues of the future work.

7.1 CONTRIBUTION

This section focuses on the contribution concerns of the DFWDAV model. It includes the problems in the literature survey models which are avoided in the DFWDAV model. Moreover, the features which are introduced or emphasized in the DFWDAV model are listed. In contrast, the features of the survey models, which are not available in the DFWDAV model, are discussed.

The significant contributions of this thesis include:

- **Proposing a systematic document flow model.**
- **Designing an agent which handles the linear document flow.**
- **Providing a proof of concept implementation of the model by applying it on a cooperative authoring system.**
- **Comparing several possible implementations.**
- **Showing the wide support of the model by open and commercial software.**

Moreover, the DFWDAV model has several features such as:

- **Offering an infrastructure for the document cooperative applications in a modular fashion.**

- **Utilizing several modern computing technologies that came into the picture due to the WebDAV protocol.**
- **Providing platform independent model.**
- **Offering systematic document flow model.**

In spite of the fact that the model is well structured, it is very flexible in several aspects including the editing modes, the document types used, the flow mechanisms, the document repositories and the supported platforms. These directions are elaborated concisely in the following:

- **The editing modes: The model supports two different possible editing modes. The first mode is the editing in place. In other words, the users can edit the document directly in the document repositories. Several clients support these editing mode as explained previously. This editing mode is suitable for reliable networks and for documents which are supported by WebDAV clients. The second mode of editing applies lock-download-update-upload-unlock methodology. In detail, the model allows the user who intend to modify a document to lock and download it so he can modify it locally by his favorite editors. When he completes his work, he uploads it back to the server and unlocks it. The locking and unlocking are necessary to prevent the lost update problem. This editing methodology is applicable with any document type and document editor, but it is more suitable for unreliable networks or with document editors which do not support WebDAV protocol.**

- **The document types used: The model does not force the users to use any specific document type. Thus, any document type can be used in the model even though the documents hold textual information or non-textual information (i.e. images, sound…ect). If the document type can be edited by a WebDAV client, the previous two editing methodologies are appropriate. In contrast, for the document types which cannot be edited by a WebDAV client, they are only applicable by the second editing methodology.**
- **The flow mechanisms: The linear flow is tackled in detail in the provided implementation. Furthermore, the model represents general document flow model which supports any flow mechanism including complicated flow types such as parallel, branching (i.e. conditional) and iterative flows. To apply such flow, the implementation should consider twofold main concerns. The first is related to the agent which is responsible for analyzing the flow definitions and therefore care should be taken for the agent implementation of complicated flow types. The second is related to the document flow definer which is used by privileged users to define flow definitions. The document flow definer should support the selected flow mechanisms in the implementation.**
- **The document repositories: Any document repository works with the model if it supports WebDAV protocol. As shown in the model implementation, two different document repositories are supported: file systems and databases. Moreover, some repositories provide more facilities such as supporting FTP and HTTP protocols and providing native XML repositories. In brief, with regard the implementation, wide possible repositories types are available.**

- **The supported platforms: As stated previously in the implementation, the model is built on the WebDAV protocol, which is widely supported by both the open and commercial software. In addition, the WebDAV software is rabidly increasing. Thus gives the implementers wide choices of suitable software over almost any platform. Moreover the model supports the Internet which increases the model independence of the platform. In sum, the model supports portable implementation.**

Furthermore, The DFWDAV model eliminates several shortages in the literature survey models. These shortages include:

- **Most of the survey document flow models are part of complicated application. This leads to handle the flow issues poorly.**
- **Most models require explicit user actions to arrange the document flow either by phone or email.**
- **Several models are tackled with either special document format or just specific public format.**

On the other hand, the DFWDAV model introduced or emphasis several new features, including:

- **Providing generic document flow frame which can be utilized by different collaborative applications.**
- **Removing the complexity of handling collaborative issues from the application side to the network layer by utilizing the WebDAV protocol which is a novel network protocol.**

- **The ability to choose the editing mode. The edit mode can be either directly on the server or locally by downloading the documents to the client where they are edited.**
- **Defining the document flows in systematic way which are handled by the routing agent.**

Finally, there are some features in the literature survey models which are not considered in the DFWDAV model because these features are not incorporated directly with the document flows. These features include:

- **The annotations capability.**
- **The BSCW event services that allows users to register for several advanced events.**

7.2 FUTURE WORK

The research presented in this thesis has been applicable for the collaborative applications and has been implemented successfully. However, further research and development are still possible in several directions, which include the flow definition, the document types and the WebDAV protocol.

It is possible to enhance the method of the flow definition by using organized flow definitions as well as graphical definitions. For example, eXchangeable Routing Language (XRL) is an emerging organized flow definition which defines flow paths using XML syntax (59), (60), (61). It enhances the model in several phases. First, since XRL uses XML which is becoming an international standard, it is possible to parse the flow definitions using the XML parsers. Second, this language allows the

users to define complex flows which include flexible sequences and include routing based on conditions (62). On the other hand, the flow definition can be improved by describing the flow paths graphically which offer a friendlier user interface.

Another direction to extend the functionality of the model is related to the document type. As shown the model can handle any document type but it is possible to enhance the model for specific kind of documents. We recommend extending the model for XML documents because XML is becoming a public standard language for the structured documents. Moreover, XML documents are capable of achieving several improvements. First, the version management can be achieved for the XML documents with quick retrieval and economical storage (63). Second, it is possible to enhance the search features for XML document by utilizing the studies related to the search of XML data contents (64), (65), (66), (67), (68). Third, XML content management systems which include XML repositories are becoming more widespread (17), (69) and so the model can be extended by employing an XML repository which ensures fast data access of the XML documents.

The final worthwhile and most crucial enhancement of the model is the entire utilizing of the WebDAV protocol. As previously made clear, the WebDAV protocol is still evolving and under construction. When it becomes a mature protocol, the model can be improved rapidly, especially in the searching and data accessing areas (70), (71).

APPENDIX A

EXAMPLES OF WEBDAV METHODS

This Appendix provides several examples related to the WebDAV methods which are mentioned in Chapter 4 for the purpose of clarifying their usages. The general structure of WebDAV methods' requests follows the format of the HTTP and comprises of the following three components (51):

- **The method: States the method to be executed by the client.**
- **Headers: Describe instructions about how the task is to be completed.**
- **A body (optional). Defines the data used in the instruction, or additional instructions, about how the method is to be executed. In the body component, XML becomes a crucial element in the overall picture of WebDAV.**

A.1 PROPERTIES

Example A.1: Retrieving Named Properties (24):

```
>>Request

PROPFIND  /file HTTP/1.1
Host: www.foo.bar
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop xmlns:R="http://www.foo.bar/boxschema/">
```

```
      <R:bigbox/>
      <R:author/>
      <R:DingALing/>
      <R:Random/>
   </D:prop>
</D:propfind>

>>Response

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
      <D:href>http://www.foo.bar/file</D:href>
      <D:propstat>
         <D:prop xmlns:R="http://www.foo.bar/boxschema/">
            <R:bigbox>
               <R:BoxType>Box type A</R:BoxType>
            </R:bigbox>
            <R:author>
               <R:Name>J.J. Johnson</R:Name>
            </R:author>
         </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
      <D:propstat>
         <D:prop><R:DingALing/><R:Random/></D:prop>
         <D:status>HTTP/1.1 403 Forbidden</D:status>
         <D:responsedescription> The user does not have access to the DingALing property.
         </D:responsedescription>
      </D:propstat>
  </D:response>
  <D:responsedescription> There has been an access violation error.
  </D:responsedescription>
</D:multistatus>
```

In Example A.1, the PROPFIND method is executed on a non-collection resource http://www.foo.bar/file. The propfind XML element specifies the name of four properties whose values are being requested. In this case only two properties were returned, since the principal issuing the request did not have sufficient access rights to see the third and fourth properties.

Example A.2: PROPPATCH (24):

```
>>Request

PROPPATCH /bar.html HTTP/1.1
Host: www.foo.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:" xmlns:Z="http://www.w3.com/standards/z39.50/>
 <D:set>
    <D:prop>
       <Z:authors>
          <Z:Author>Jim Whitehead</Z:Author>
          <Z:Author>Roy Fielding</Z:Author>
       </Z:authors>
    </D:prop>
 </D:set>
 <D:remove>
    <D:prop><Z:Copyright-Owner/></D:prop>
 </D:remove>
</D:propertyupdate>

>>Response

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
xmlns:Z="http://www.w3.com/standards/z39.50>
 <D:response>
    <D:href>http://www.foo.com/bar.html</D:href>
    <D:propstat>
       <D:prop><Z:Authors/></D:prop>
       <D:status>HTTP/1.1 424 Failed Dependency</D:status>
    </D:propstat>
    <D:propstat>
       <D:prop><Z:Copyright-Owner/></D:prop>
       <D:status>HTTP/1.1 409 Conflict</D:status>
    </D:propstat>
    <D:responsedescription> Copyright Owner can not be deleted or altered.
    </D:responsedescription>
 </D:response>
</D:multistatus>
```

In Example A.2, the client requests the server to set the value of the http://www.w3.com/standards/z39.50/Authors property, and to remove the property

http://www.w3.com/standards/z39.50/Copyright-Owner. Since the Copyright-Owner property cannot be removed, no property modifications occur. The 424 (Failed Dependency) status code for the Authors property indicates this action would have succeeded if it had not conflicted with removing the Copyright-Owner property.

A.2 COLLECTIONS AND NAMESPACE OPERATIONS

Example A.3: MKCOL (24):

```
>>Request

MKCOL /webdisc/xfiles/ HTTP/1.1
Host: www.server.org

>>Response

HTTP/1.1 201 Created
```

Example A.3 creates a collection called /webdisc/xfiles/ on the server www.server.org.

Example A.4: DELETE (24):

```
>>Request

DELETE  /container/ HTTP/1.1
Host: www.foo.bar

>>Response

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
<?xml version="1.0" encoding="utf-8" ?>
<d:multistatus xmlns:d="DAV:">
  <d:response>
      <d:href>http://www.foo.bar/container/resource3</d:href>
      <d:status>HTTP/1.1 423 Locked</d:status>
  </d:response>
</d:multistatus>
```

In Example A.4 the attempt to delete http://www.foo.bar/container/resource3 failed because it is locked, and no lock token was submitted with the request. Consequently, the attempt to delete http://www.foo.bar/container/ failed too. The client knows that the attempt to delete http://www.foo.bar/container/ had failed since the parent cannot be deleted unless its child has also been deleted.

Example A.5: COPY with Overwrite (24):

```
>>Request

COPY /~fielding/index.html HTTP/1.1
Host: www.ics.uci.edu
Destination: http://www.ics.uci.edu/users/f/fielding/index.html

>>Response

HTTP/1.1 204 No Content
```

Example A.5 shows resource http://www.ics.uci.edu/~fielding/index.html being copied to the location http://www.ics.uci.edu/users/f/fielding/index.html. The 204 (No Content) status code indicates the existing resource at the destination was overwritten.

Example A.6: COPY of a Collection (24):

```
>>Request

COPY /container/ HTTP/1.1
Host: www.foo.bar
Destination: http://www.foo.bar/othercontainer/
Depth: infinity
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<d:propertybehavior xmlns:d="DAV:">
```

```
  <d:keepalive>*</d:keepalive>
</d:propertybehavior>

>>Response

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<d:multistatus xmlns:d="DAV:">
  <d:response>
      <d:href>http://www.foo.bar/othercontainer/R2/</d:href>
      <d:status>HTTP/1.1 412 Precondition Failed</d:status>
  </d:response>
</d:multistatus>
```

In Example A.6, the Depth header is unnecessary as the default behavior of COPY on a collection is to act as if a "Depth: infinity" header had been submitted. Also in this example most of the resources, along with the collection, were copied successfully. However the collection R2 failed, most likely due to a problem with maintaining the liveness of properties (this is specified by the property behavior XML element). Because there was an error in copying R2, none of R2's members were copied. However, no errors were listed for those members due to the error minimization rules.

Example A.7: MOVE of a Non-Collection (24):

```
>>Request

MOVE /~fielding/index.html HTTP/1.1
Host: www.ics.uci.edu
Destination: http://www.ics.uci.edu/users/f/fielding/index.html

>>Response

HTTP/1.1 201 Created
Location: http://www.ics.uci.edu/users/f/fielding/index.html
```

Example A.7: shows resource http://www.ics.uci.edu/~fielding/index.html being moved to the location http://www.ics.uci.edu/users/f/fielding/index.html. The contents of the destination resource would have been overwritten if the destination resource had been non-null. In this case, since the operation was successful, the response code is 201 (Created).

Example A.8: MOVE of a Collection (24):

```
>>Request

MOVE /container/ HTTP/1.1
Host: www.foo.bar
Destination: http://www.foo.bar/othercontainer/
Overwrite: F
If: (<opaquelocktoken:fe184f2e-6eec-41d0-c765-01adc56e6bb4>)
   (<opaquelocktoken:e454f3f3-acdc-452a-56c7-00a5c91e4b77>)
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<d:propertybehavior xmlns:d='DAV:'>
  <d:keepalive>*</d:keepalive>
</d:propertybehavior>

>>Response

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<d:multistatus xmlns:d='DAV:'>
  <d:response>
      <d:href>http://www.foo.bar/othercontainer/C2/</d:href>
      <d:status>HTTP/1.1 423 Locked</d:status>
  </d:response>
</d:multistatus>
```

In Example A.8 the client has submitted a number of lock tokens with the request. A lock token will need to be submitted for every resource, both source and destination, anywhere in the scope of the method, that is locked. In this case the

proper lock token was not submitted for the destination http://www.foo.bar/othercontainer/C2/. This means that the resource /container/C2/ could not be moved. Because there was an error copying /container/C2/, none of /container/C2's members were copied. However no errors were listed for those members due to the error minimization rules. User agent authentication has previously occurred via mechanism outside the scope of the HTTP protocol, in an underlying transport layer.

A.3 LOCKING

Example A.9 Simple Lock Request (24):

```
>>Request

LOCK /workspace/webdav/proposal.doc HTTP/1.1
Host: webdav.sb.aol.com
Timeout: Infinite, Second-4100000000
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ejw",
  realm="ejw@webdav.sb.aol.com", nonce="...",
  uri="/workspace/webdav/proposal.doc",
  response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:lockinfo xmlns:D='DAV:'>
 <D:lockscope><D:exclusive/></D:lockscope>
 <D:locktype><D:write/></D:locktype>
 <D:owner>
     <D:href>http://www.ics.uci.edu/~ejw/contact.html</D:href>
 </D:owner>
</D:lockinfo>

>>Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
```

```
<D:prop xmlns:D="DAV:">
 <D:lockdiscovery>
    <D:activelock>
       <D:locktype><D:write/></D:locktype>
       <D:lockscope><D:exclusive/></D:lockscope>
       <D:depth>Infinity</D:depth>

       <D:owner>
          <D:href>
             http://www.ics.uci.edu/~ejw/contact.html
          </D:href>
       </D:owner>
       <D:timeout>Second-604800</D:timeout>
       <D:locktoken>
          <D:href>
       opaquelocktoken:e71d4fae-5dec-22d6-fea5-00a0c91e6be4
          </D:href>
       </D:locktoken>
    </D:activelock>
 </D:lockdiscovery>
</D:prop>
```

Example A.9 shows the successful creation of an exclusive write lock on resource http://webdav.sb.aol.com/workspace/webdav/proposal.doc. The resource http://www.ics.uci.edu/~ejw/contact.html contains contact information for the owner of the lock. The server has an activity-based timeout policy in place on this resource, which causes the lock to automatically be removed after one week (604800 seconds). The nonce, response, and opaque fields have not been calculated in the Authorization request header.

Example A.10 UNLOCK (24):

```
>>Request

UNLOCK /workspace/webdav/info.doc HTTP/1.1
Host: webdav.sb.aol.com
Lock-Token: <opaquelocktoken:a515cfa4-5da4-22e1-f5b5-00a0451e6bf7>
Authorization: Digest username="ejw",
  realm="ejw@webdav.sb.aol.com", nonce="...",
  uri="/workspace/webdav/proposal.doc",
  response="...", opaque="..."
```

>>Response

HTTP/1.1 204 No Content

In example A.10, the lock identified by the lock token "opaquelocktoken:a515cfa4-5da4-22e1-f5b5-00a0451e6bf7" is successfully removed from the resource http://webdav.sb.aol.com/ workspace/webdav/info.doc. If this lock included more than just one resource, the lock is removed from all resources which included in the lock. The 204 (No Content) status code is used instead of 200 (OK) because there is no response entity body. In this example, the nonce, response, and opaque fields have not been calculated in the Authorization request header.

A.4 VERSIONING

Example A.11: VERSION –CONTROL (51):

>>REQUEST

VERSION-CONTROL /file.html HTTP/1.1
Host: www.webdav.org

>>RESPONSE

HTTP/1.1 200 OK

In example A.11, /file.html is put under version control. A new version history is created for it, and a new version is created that has a copy of the content and dead properties of /file.html. The DAV:checked-in property of /file.html identifies this new version.

Example A.12: CHECKOUT (51):

>>REQUEST

```
CHECKOUT / file.html HTTP/1.1
Host: www.webdav.org
```

>>RESPONSE

```
HTTP/1.1 200 OK
```

Example A.3: CHECKIN (51):

>>REQUEST

```
CHECKIN / file.html HTTP/1.1
Host: www.webdav.org
```

>>RESPONSE

```
HTTP/1.1 201 Created
Location: http://repo.webdav.org/his/23/ver/32
```

Example A.15: UNCHECKOUT (51):

>>REQUEST

```
UNCHECKOUT / file.html HTTP/1.1
Host: www.webdav.org
```

>>RESPONSE

```
HTTP/1.1 200 OK
```

Example A.16: Report with DAV:available-report (51):

>>REQUEST

```
REPORT /myCollection HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<DAV:available-report/>
```

>>RESPONSE

```
HTTP/1.1 200 OK
Host: www.webdav.org
```

```
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:report-set xmlns:D="DAV:">
  <D:available-report/>
  <D:property-report/>
</D:report-set>
```

Example A.16 uses the DAV:available-report, which lists the reports supported at the request-URL.

Example A.17: Report with DAV:property-report (51):

```
>>REQUEST

REPORT /file HTTP/1.1
Host: www.webdav.org
Target-Selector: versioned-resource
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:property-report xmlns:D="DAV:">
  <D:revisions>
    <D:revision-id/>
    <D:author/>
  </D:revisions>
</D:property-report>

>>RESPONSE

HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/file</D:href>
    <D:propstat>
      <D:prop>
        <D:revisions>
          <D:response>
            <D:href>http://repo/rev/id182<D:href>
            <D:propstat>
              <D:prop>
                <D:revision>id182</D:revision>
                <D:author>Fred</D:author> </D:prop>
              <D:status>HTTP/1.1 200 OK</D:status>
```

```
      </D:propstat> </D:response>
     <D:response>
      <D:href>http://repo/rev/id263<D:href>
      <D:propstat>
       <D:prop>
        <D:revision>id263</D:revision>
        <D:author>Sally</D:author> </D:prop>
       <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat> </D:response>
    </D:revisions> </D:prop>
   </D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat> </D:response>
 </D:multistatus>
```

A.5 ACCESS CONTROL

Example A.18: Retrieving Access Control information (52):

```
>>Request

PROPFIND /top/container HTTP/1.1
Host: www.foo.bar
Content-type: text/xml; charset="utf-8"
Content-Length: 0
Depth: 0

<?xml version=”1.0”>
<D:propfind xmlns:D=”DAV:”>
  <D:allprop/>
</D:propfind>

>>Response

HTTP/1.1 200 Success
Content-Type: text/xml
Content-Length: xxxxx

<?xml version="1.0" encoding="utf-8" ?>
<?namespace href = “http://www.ietf.org/standards/webdav/ AS = D"?>
<D:response>
  <D:propstat>
    <D:creationdate>1997-12-01T17:42:21-08:00</D:creationdate>
    <D:displayname>Example collection</D:displayname>
    <D:resourcetype><D:collection/></D:resourcetype>
    <D:supportedlock> XXXXX</D:supportedlock>
    <D:owner>http://www.rational.com/principals/users/gclemm</d:owner>
    <D:owner-name>Geoffrey Clemm</d:owner-name>
    <D:rights>
      <D:read/><D:readacl/>
```

```
  </D:rights>
  <D:acl>
        <D:ace>
          <D:grant><D:read/><D:write/><D:readacl/></D:grant>
          <D:principal-id>
   <D:href>http://www.foo.com/users/esedlar</D:href>
  </D:principal-id>
          <D:principal>
            <D:principal-type>User</D:principal-type>
            <D:principalname>esedlar</D:principalname>
            <D:displayname>Eric Sedlar</D:displayname>
          </D:principal>
        </D:ace>
        <D:ace>
          <D:grant><D:read/><D:readacl/></D:grant>
          <D:deny><D:writeacl/></D:deny>
          <D:principal-id>
            <D:href>http://www.foo.com/groups/marketing</d:href>
  </D:principal-id>
          <D:principal>
            <D:principal-type>Group</D:principal-type>
            <D:displayname>Foo.Com marketing department</D:displayname>
            <D:principalname>mktdept</D:principalname>
          </D:principal>
        </D:ace>
        <D:ace>
          <D:grant><D:read/></D:grant>
          <D:principal-id><d:all/></D:principal-id>
        </D:ace>
   </D:acl>
  <D:propstat>
<D:response>
```

Example A.19: Setting ACLs (52):

```
>>Request

ACL /top/container HTTP/1.1
Host: www.foo.com
Content-Type: text/xml
Content-Length: xxxx

<?namespace href = "http://www.ietf.org/standards/webdav/ AS = D"?>
<D:acl-info>
  <D:acl>
        <D:ace>
           <D:grant><D:read/><D:write/><D:readacl/></D:grant>
           <D:principal-id>
              <D:href>http://www.foo.com/users/esedlar</D:href>
          </D:principal-id>
        </D:ace>
```

```
        <D:ace>
          <D:grant><D:read/> </D:grant>
          <D:principal-id>
             <D:href>http://www.foo.com/groups/marketing</D:href>
          </D:principal-id>
        </D:ace>
  </D:acl>
</D:acl-info>

>>Response

HTTP/1.1 200 Success
Content-Length: 0
```

In example A.19, an ACL request body is an acl-info XML element. The <dav:acl-info> element contains properties that can be set by the ACL method (currently just <acl>). This example changes the group ACE to disallow read access to the ACL for the marketing group.

A.6 SEARCHING

Example A.20: Query Example (53):

```
<d:searchrequest>
  <d:basicsearch>
    <d:select>
      <d:prop><d:getcontentlength/></d:prop>
    </d:select>
    <d:from>
      <d:scope>
        <d:href>/container1/</d:href>
        <d:depth>infinity</d:depth>
      </d:scope>
    </d:from>
    <d:where>
      <d:gt>
        <d:prop><d:getcontentlength/></d:prop>
        <d:literal>10000</d:literal>
      </d:gt>
    </d:where>
      <d:orderby>
        <d:order>
        <d:prop><d:getcontentlength/><d:prop>
        <d:ascending/>
```

```
    </d:order>
   </d:orderby>
  </d:basicsearch>
</d:searchrequest>
```

Example A.20 provides a query which retrieves the content length values for all resources located under the server's "/container1/" URI namespace whose length exceeds 10000.

Nomenclature

ACE	**Access Control Entry**
ACL	**Access Control Lists**
API	**Application Program Interface**
BSCW	**Basic Support for Cooperative Work**
CSS	**Cascading Style Sheets**
DASL	**DAV Searching and Locating**
DAV	**Distributed Authoring and Versioning**
DAV4J	**DAV for Java**
DFWDAV	**Document Flow model based on WebDAV protocol**
DMS	**Document Management Systems**
DOM	**Documents Object Model**
DReSS	**Document Repository Service Station**
DTD	**Document Type Definition**
EDMS	**Engineering Document Management System**
ER	**Entity Relationship**
GML	**General Markup Language**
HTML	**Hypertext Markup Language**
HTTP	**Hypertext Transfer Protocol**
ISO	**International Organization for Standardization**

IETF	**Internet Engineering Task Force**
IFS	**Oracle Internet File System**
IIS	**Microsoft Internet Information Services**
KQML	**Knowledge Query and Manipulation Language**
SAX	**Simple API for XML**
SGML	**Standard Generalized Markup Language**
SIM	**Structured Information Manager**
SMTP	**Simple Mail Transfer Protocol**
W^3C	**World Wide Web Consortium**
WAN	**Wide Area Network**
WebDAV	**World Wide Web Distributed Authoring and Versioning protocol**
WWW	**World Wide Web**
XML	**eXtensible Markup Language**
XRL	**eXchangeable Routing Language**
XSL	**eXtensible Style Language**

Bibliography

1 Document Management Avenue Frequently Answered Questions V1.1, http://www.documentmanagement.org.uk/pages/faq.htm, Sep. 2001.

2 R. Sprague, "Electronic Document Management: Challenges and Opportunities for Information Systems Managers", *MIS Quarterly*, Mar. 1995.

3 F. Bapst and C. Vanoirbeek, "XML Documents Production for an Electronic Platform of Requests For Proposals", *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, 330-335, 1999.

4 P. Ciancarini, F. Vitali and C. Mascolo, "Managing Complex Documents Over the WWW: A Case Study for XML", *IEEE Transaction on Knowledge and Data Engineering*, Vol. 11, No. 4, 629-638, July/August 1999.

5 H. Kanemoto, H. Kato, H. Kinutani and M. Yoshikawa, "An Efficiently Updatable Index Scheme for Structured Documents", *The 9th International IEEE Workshop on Database and Expert Systems Applications (DEXA'98)*, 991-996, Aug. 1998.

6 D. Rossi and F. Vitali, "Internet-Based Coordination Environments and Document-Based Applications: a Case Study", *Proceedings: The International Conference on Coordination Models and Languages*, 259-274, Apr. 1999.

7 R. Summers, J. J. L. Chelsom, D. R. Nurse and J. D. S. Kay, "Document Management – an Intranet Approach", *18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1236-1237 vol.3, 1997

8 D. Sussman, "WebDAV: A Panacea for Collaborative Authoring?", *IEEE Multimedia*, Vol. 6 2, 76-79, April/June 1999.

9 E. Whitehead and M. Wiggings, "WebDAV: IETF Standard for Collaborative Authoring on the Web", *IEEE Internet Computing*, Vol. 2 5, 34-40, September/October 1998.

10 I. Sengupata, "Toward the Union of Database and Document Management: The Design of DocBase", *Proceedings: Conference on Management of Data (COMAD'98)*, Dec. 1998.

11 A. Backer and U. Busbach, "DocMan: A Document Management System for Cooperation Support", *Proceedings of the 29th IEEE International Conference on System Sciences*, 82-91, 1996.

12 J. Robie, "XML and Modern Software Architectures", *SGML/XML '97*, http://www.texcel.no/sgml97.tml, 1997.

13 H. Lie and J. Saarela, "Multipurpose Web Publishing Using HTML, XML and CCS", *Communications of The ACM*, Vol. 42, No. 10, 95-101, Oct. 1999.

14 N. Pitts, *XML in Record Time*, SYBEX, 1999.

15 T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, "Extensible Markup Language, (XML) 1.0 (Third Edition)", W3C Recommendation 4-February-2004, http://www.w3.org/TR/2004/REC-xml-20040204.

16 E. Xavier P., "Applying XML Architecture to Web Applications", White Paper, Infosys Technologies Ltd., Apr. 2000. http://www.inf.com/corporate/thought-papers/XML_April20.htm

17 T. Arnold-Moore, M. Fuller, A. Kent, R. Sacks-Davis, and N. Sharman, "Architecture of a Content Management Server for XML Document Applications", *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE2000)*, Hong Kong, June 19-20, 2000.

18 C. Yang, S. Ju and T. Rao, "A Smartcard-based Framework for Secure Document Exchange", *Proceedings of the 32nd IEEE International Carnahan Conference on Security Technology*, 93-96, 1998.

19 S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman and S. Zilles, "Extensible Stylesheet Language (XSL) Version 1.0", W3C Recommendation 15 October 2001, http://www.w3.org/TR/2001/REC-xsl-20011015.

20 B. Bos, H. Lie, C. Lilley and I. Jacobs, "Cascading Style Sheets, level 2 CSS2 Specification", W3C Recommendation 12-May-1998, http://www.w3.org/TR/1998/REC-CSS2-19980512

21 H. Lie and B. Bos, "Cascading Style Sheets, level 1", W3C Recommendation revised 11-Jan-1999, http://www.w3.org/TR/1999/REC-CSS1-19990111

22 V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, T. Research, R. Sutor, C. Wilson and L. Wood, "Document Object Model, (DOM) 1.0", W3C Recommendation 1-October-1998, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001

23 A. Hors, P. Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion and S. Byrne, "Document Object Model, (DOM) Level 2 Core Specification Version 1.0", W3C Recommendation 13-November-2000, http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113

24 Y. Goland, E. Whitehead, A. Faizi, S. Carter and D. Jensen, "HTTP Extensions for Distributed Authoring", RFC2518, IETF, Feb. 1999.

25 E. Whitehead, "World Wide Web Distributed Authoring and Versioning (WEBDAV): An Introduction", *ACM StandardView*, Vol. 5, No. 1, 3-8, March 1997.

26 F. Dridi and G. Neumann, "How to implement Web-based Groupware Systems based on WebDAV", *Proceedings: International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'99)*, 114-119, 1999.

27 M. Adkins, J. Reinig, J. Kruse and D. Mittleman, "GSS Collaboration in Document Development: Using GroupWriter to Improve the Process", *Thirty - Second Annual Hawaii International Conference on System Sciences*, Maui, Hawaii, 5 - 8 January, 1999.

28 D. Decouchant, A. Enríquez and E. González, "AllianceWeb: Cooperative Authoring on the WWW", *Proceedings of the String Processing and Information Retrieval Symposium & International Workshop on Groupware*, September 21-24, 1999, Cancun, Mexico.

29 T. Horstmann and R. Bentley, "Distributed Authoring on the Web with the BSCW Shared Workspace System", *ACM Standards View* 5(1), March 1997.

30 H. Peltonen, "EDMS engineering data management system—architecture and concepts", Helsinki University of Technology, *Laboratory of Information Processing Science*, 1993.

31 H. Peltonen, T. Männistö, K. Alho and R. Sulonen, "An Engineering Document Management System", *The Winter Annual Meeting of the American Society of Mechanical Engineers*, Paper 93-WA/EDA-1, 1993.

32 D. Decouchant, V. Quint and M. Romero Salcedo, "Structured Cooperative Authoring on the World Wide Web", *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, USA, December 11-14, 1995.

33 D. Decouchant and M. Romero Salcedo, "Alliance: A Structured Cooperative Editor on the Web*", Proceedings of the ERCIM workshop on CSCW and the Web*, Germany, February 7-9, 1996.

34 P. De Bra and A. Aerts, "Multi-User Publishing in the Web: DReSS, A Document Repository Service Station", *Proceedings of the ERCIM workshop on CSCW and the Web*, Sankt Augustin, Germany, February 7-9, 1996

35 A. Aerts, P. De Bra and M. Timmermans, "DReSS 2.0: Lightweight groupware for hypertext publishing on the Web", *Proceedings of the AACE WebNet'98 Conference*, pp. 20-25, Orlando, Fl., 1998.

36 R. Bentley and W. Appelt, "Designing a System for Cooperative Work on the World-Wide Web: Experiences with the BSCW System", *Proceedings of HICSS'30: The Hawaii International Conference on the System Sciences*, Maui, Hawaii, January 7-10, 1997.

37 R. Bentley, T. Horstmann and J. Trevor, "The World Wide Web as enabling technology for CSCW: The case of BSCW", *Computer Supported Cooperative Work: The Journal of Collaborative Computing*. Special issue on CSCW and the Web, Vol. 6, 1997.

38 W. Appelt, "WWW Based Collaboration with the BSCW System", *Proceedings of SOFSEM'99*, *Springer Lecture Notes in Computer Science*, p.66-78; November 26 - December 4, Milovy, Czech Republic.

39 J. Kurose and K. Ross, *Computer Networking A Top-Down Approach Featuring the Internet*, Addison-Wesley, August 2000, http://www.awlonline.com/kurose/

40 T. Arnold-Moore, *Information systems for legislation*, Doctoral Thesis, Royal Melbourne Institute of Technology, Melbourne, 1998.

41 T. Arnold-Moore, M. Fuller and R. Sacks-Davis, "Approaches for Structured Document Management", *Markup Technologies '99*, Philadelphia, PA, U.S.A, December 7-9,1999.

42 *EnAct: User's Guide*, The Quill Consultancy Pty Ltd, Australia, Jul. 1999.

43 M. Petterson and G. Lysén, *Document Management within Intranet Settings*, Master Thesis, Department of Computer Science, Lund Institute of Technology, Swede, 1999.

44 R. Guetari, V. Quint and I. Vatton, "Amaya: an Authoring Tool for the Web", *MCSEAI'98 International Conference*, Tunisie, December 1998.

45 J. Cadiz, A. Gupta and J. Grudin, "Using Web Annotations for Asynchronous Collaboration Around Documents", *CSCW 2000 ACM 2000 Conference on Computer Supported Cooperative Work*, Philadelphia, Pennsylvania, USA, December 2 - 6, 2000.

46 P. Juell and M. Habib, "A Protocol to Develop Agent-Based Form Flow Systems", *The 37th Annual Midwest Instruction and Computing Symposium (MICS04)*, University of Minnesota, Morris, 16-17th April 2004.

47 D. Rossi, "Orchestrating document-based workflows with X-Folders", *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*, Nicosia, Cyprus, 14-17 March 2004, pp.503-507.

48 J. Slein, F. Vitali, E. Whitehead and D. Durand, "Requirements for Distributed Authoring and Versioning Protocol for the World Wide Web", RFC2291, IETF, Feb. 1998.

49 WWW Distributed Authoring and Versioning (WebDAV) Home Page, http://www.ietf.org/html.charters/webdav-charter.html , Feb. 2005.

50 E. Whitehead, IETF WEBDAV Working Group Home Page, http://www.ics.uci.edu/pub/ietf/webdav/, Jul 2003.

51 G. Clemm, J. Amsden, C. Kaler and J. Whitehead, "Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning))", RFC3253, IETF, Mar. 2002.

52 G. Clemm, J. Reschke, E. Sedlar and J. Whitehead, "WebDAV Access Control Protocol", RFC3744**,** IETF, May 2004.

53 J. Reschke, S. Reddy, J. Davis and A. Babich, "WebDAV Search", Internet-Draft, IETF, Feb. 2005.

54 G. Stein, WebDAV Resources, http://www.webdav.org/ , Sep. 2004.

55 The Apache HTTP Server Project, http://httpd.apache.org/, 2004.

56 Jigsaw - W3C's Server, http://www.w3.org/Jigsaw/, Nov. 2004.

57 Slide, http://jakarta.apache.org/slide/index.html, 2004.

58 DAV Explorer, http://www.ics.uci.edu/~webdav/, Feb. 2005.

59 W. Aalst and A. Kumar, "XML Based Schema Definition for Support of Inter-organizational Workflow.", *21st International Conference on Application and Theory of Petri Nets (ICATPN 2000)*, Aarhus, Denmark, June 26-30, 2000.

60 A. Kumar and J. Zhao, "XRL: An Interoperable Routing Standard for E-Commerce Applications", *International Workshop on Component-based Electronic Commerce*, Berkerley, California, July 25, 1998.

61 A. Kumar and J. Zhao, "Dynamic Routing and Operational Controls in Workflow Management Systems", *Management Science*, Vol. 45, No. 2, 253-272, Feb 1999.

62 A. Kumar and J. Zhao, "XRL: An Extensible Routing Language for Electronic Commerce Applications", *Proceedings of the First International Conference on Telecommunications and Electronic Commerce*, Nashville, USA, Nov 20-22, 1998.

63 S. Chien, V. Tsotras and C. Zaniolo, "Version Management of XML Documents*", WebDB 2000 Workshop*, Dallas, TX, 2000.

64 S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie and J. Siméon, "XQuery: An XML Query Language", W3C Working Draft 11 February 2005, http://www.w3.org/TR/2005/WD-xquery-20050211/.

65 K. Böhm, "On Extending the XML Engine with Query-Processing Capabilities", *Proc. of IEEE Advances in Digital Libraries*, Bethesda, Maryland, May, 2000.

66 S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi and L. Tanca, "XML-GL: a graphical language for querying and restructuring XML documents", *Computer Networks*, Vol. 31, No. 11-16, 1171-1187, 1999.

67 J. Miller and S. Sheth, "Querying XML documents", *IEEE Potentials*, Volume 19, Issue 1, 24 –26, Feb.-March 2000.

68 A. Deutsch, M. Fernandez, D. Florescu, A. Levy and D. Suciu "A Query Language for XML", *Computer Networks* 31(11-16), 1999.

69 T. Freter, "XML: Document and Information Management", Sun Inc., http://www.sun.com/980908/xml/, Sep. 1998.

70 S. Kim, K. Pan, E. Sinderson and E. Whitehead, "Architecture and Data Model of a WebDAV-based Collaborative System", *Proceedings of The 2004 International Symposium on Collaborative Technologies and Systems (CTS 2004)*, San Diego, California, 18-23 January 2004.

71 S. Kim, M. Slater, E. Whitehead, "WebDAV-based Hypertext Annotation and Trail System", *Proceedings of the Fifteenth ACM Confereence on Hypertext and Hypermedia (Hypertext 2004),* Santa Cruz, California, pp. 87-88, 9-13 August 2004.

Vita

Personal information

Name: Hussam Eddin Abdullah Al-Sawadi
Date of birth: 1971
Email: hussam@ccse.kfupm.edu.sa
and
hussamalsawadi@hotmail.com
Current Position: Information Technology Expert in Ministry of Justice

Qualifications

June 2005: **M.Sc. degree in information and computer science** with high honor (G.P.A. of 3.91/4.00).
Information and Computer Science Department, College of Graduate Studies, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

July 1993: **B.Sc. degree in computer engineering** with high honor (G.P.A. of 4.64/5.00).
Department of Computer Engineering, Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia.

Research experience

1995: **Advanced Arabic Authoring Language (CATIB) for Computer Assisted Instructions (CAI).** King Abdul Aziz City for Science and Technology (KACST)

1993: **Arabic character recognition using Backpropagation Neural Network**. King Saud University (KSU).