

## **INFORMATION TO USERS**

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

**UMI**<sup>®</sup>  
800-521-0600



DESIGN OF A USER INTERFACE TO FACILITATE SEARCHING  
IN A DIGITAL LIBRARY

KAVITHA PATHY

A THESIS  
IN  
THE DEPARTMENT  
OF  
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE  
CONCORDIA UNIVERSITY  
MONTRÉAL, QUÉBEC, CANADA

DECEMBER 1999

© KAVITHA PATHY, 2000



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-48304-5

Canada

# Abstract

Design of a User Interface to facilitate searching in a Digital Library

Kavitha PATHY

The widespread acceptance of digital library will depend, to a large extent, on the ease-of-use and appeal of the user interface. Hence, the development of a versatile and friendly user interface for digital library becomes indispensable and this has been the objective of this thesis. A pilot project on digital library was undertaken jointly by the four universities in Montreal. At Concordia University we undertook the responsibility for developing a User Interface. The sample digital library consists of 9000 photographs of architectural images and associated texts. The user interface design deals with providing a uniform interface for texts and images. It makes use of a simple conceptual model that is well suited for digital libraries, and provides a direct manipulation interface with affordances that are appropriate to a "distributed digital library". The biggest challenge in a distributed digital library will be to balance between recall and precision to a level satisfactory to the user. Most searches on digital library often provide a large number of hits and thus overloading the user with information. To overcome the problem of information overloading, in this thesis, we have investigated collaboration in information retrieval. Collaboration among multiple users stem from reusing and sharing information gathered by one with others. Software agents called user interface agents are expected to assist users in collaboration and filtering. A protocol is proposed for cooperation and inter-operation between multiple user interface agents. The analysis shows that conflicts can arise during collaboration and they can be resolved using negotiation.

A prototype of the user interface has been developed in Java and has been integrated into the joint project from other universities. Usability testing and gathering user feedback remains to be done in the future.

# Acknowledgments

My deepest gratitude goes to my supervisors, Dr. T. Radhakrishnan and Dr. R. Shinghal. Dr. T. Radhakrishnan has been and will continue for many years to be a role model for me, on both professional and personal levels.

I would like to acknowledge the advice and feedback received from the members of the Digital Library research group. I would also like to thank Newbridge for funding this project.

I would like to thank Venkatachalam, Bhasker and the members of multimedia lab for friendship, good advice and moral support. I would also like to thank people outside of the multimedia lab in whose company I found many diversions.

Thanks to administrative staff of the department of computer science, to Pauline Dubois and Halina Monkiewicz. I own special thanks to the technical staffs Michael Spanner, Stan Swiercz and Caroline Hakim for their timely help through out my thesis.

Special thanks to my parents for making me what I am today, and to my brother and husband for believing in me and supporting me. Finally, thank you Vignesh for making me feel so special.

# Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	1
1.2 Context: Digital Library . . . . .	3
1.3 Need Analysis . . . . .	5
1.3.1 Motivation . . . . .	5
1.3.2 Goals . . . . .	6
1.4 Contribution of this thesis . . . . .	7
<b>2 Literature Survey</b>	<b>9</b>
2.1 What is an Agent . . . . .	9
2.2 Classification of agents . . . . .	10
2.2.1 Collaborative agents . . . . .	11
2.2.2 Interface Agents . . . . .	11
2.2.3 Mobile Agents . . . . .	13
2.2.4 Information Agents . . . . .	14

2.3	Agent Modeling . . . . .	15
2.3.1	Agent Theory Approaches . . . . .	15
2.3.2	Architectures . . . . .	16
2.3.3	Communication . . . . .	17
2.4	Application of agent Technology in Digital libraries . . . . .	17
2.4.1	University of Michigan Digital Library (UMDL) . . . . .	18
2.4.2	University of Sunderland Digital Library . . . . .	22
2.5	Summary . . . . .	27
<b>3</b>	<b>User Interface for Digital Library</b>	<b>29</b>
3.1	Architecture . . . . .	30
3.1.1	Registry subsystem . . . . .	30
3.1.2	Query subsystem . . . . .	31
3.1.3	Filter Subsystem . . . . .	32
3.2	Visual Design . . . . .	35
3.2.1	Requirements . . . . .	35
3.2.2	Visual Design of the Prototype . . . . .	37
3.3	User Interactions . . . . .	37
3.3.1	Affordances . . . . .	38
3.3.2	Feedback mechanisms . . . . .	41
3.4	Class Diagrams . . . . .	42
3.5	Object Interaction Diagram . . . . .	47



<b>4</b>	<b>Cooperative navigation in digital library</b>	<b>50</b>
4.1	System Design . . . . .	51
4.1.1	Agent . . . . .	52
4.1.2	The Agent Language . . . . .	54
4.1.3	System Architecture . . . . .	57
4.2	Functions of the system . . . . .	59
4.2.1	Navigation in the site-document graph . . . . .	59
4.2.2	Case study . . . . .	61
4.2.3	Algorithm for cooperative navigation (Case - 1: two agents situation.)	63
4.2.4	Detecting conflict and resolving conflicts by negotiation . . . . .	65
4.3	Summary . . . . .	68
<b>5</b>	<b>Implementation Overview</b>	<b>69</b>
5.1	Implementation of the User interface . . . . .	69
5.1.1	Registering the user . . . . .	70
5.1.2	Queries formulation . . . . .	71
5.1.3	Retrieving the search results . . . . .	72
5.1.4	Filtering and displaying the search results . . . . .	74
5.2	Description of the GUI . . . . .	76
5.2.1	Response-set presentation . . . . .	77
<b>6</b>	<b>Conclusion</b>	<b>83</b>
6.1	Goals revisited . . . . .	83
6.1.1	Support user's tasks . . . . .	84
6.1.2	Sharing and reusing information gathered . . . . .	84
6.2	Discussion . . . . .	85



# List of Figures

1.1	System Architecture (TCP/IP: Transmission Control Protocol/Internet Protocol; API: Application programming interface) . . . . .	4
2.1	Functioning of Interface Agents. . . . .	12
2.2	UMDL agent types . . . . .	19
2.3	Hypermedia Agent Architecture . . . . .	23
2.4	A Transducer . . . . .	23
2.5	A Model for distributed information retrieval . . . . .	26
2.6	Federated System . . . . .	27
3.1	System architecture with two alternative data flow sequences . . . . .	30
3.2	RPN query represented by tree . . . . .	33
3.3	Main Class Diagram . . . . .	43
3.4	Class Diagram for Attribute Panel . . . . .	44
3.5	Class Diagram for panell . . . . .	44
3.6	Class Diagram for Similarity panel . . . . .	45
3.7	Class Diagram for Result panel . . . . .	46
3.8	Initial object interaction diagram . . . . .	48
3.9	Detailed object interaction diagram (when Total number of documents is less than Theta) . . . . .	49

4.1	System framework for cooperative navigation. <i>The dotted line shows the communication between agents. All agents communicate with the registry, which is shown by a solid line.</i> . . . . .	51
4.2	System architecture . . . . .	52
4.3	BDI architecture . . . . .	53
4.4	Site-Document graph . . . . .	60
4.5	Site-Document(Marked) graph . . . . .	61
4.6	Site-Document graph $G_1$ for query $Q_1$ . <i>Circles represents the different sites and the arch represents the link between them, and the number within the bracket represents number of documents in the site which match the query.</i> .	62
4.7	Protocol diagram for two agent situation . . . . .	64
4.8	Communication between agents . . . . .	66
4.9	Protocol for three agents situation. <i>In this <math>A_1</math> gets a proposal from <math>A_2</math> and <math>A_3</math>. It then compares the proposal. Since it is not equal, it uses compromise act once with <math>A_3</math> to find the best proposal.</i> . . . . .	67
4.10	Directed graph showing the confidence function. <i>The weightage is between 1-10. The number in the graph represents the weight of confidence.</i> . . . . .	67
5.1	Functional model for registration . . . . .	70
5.2	Functional model for verifying the password . . . . .	70
5.3	Functional model for retrieving the search results . . . . .	73
5.4	Functional model for filtering . . . . .	74
5.5	Functional model for displaying the URLs sequentially without filtering . .	75
5.6	Functional model for displaying the URLs sequentially . . . . .	75
5.7	Opening window of the Digital Library . . . . .	78
5.8	Interface for attribute based search . . . . .	79

5.9	Interface for similarity based search . . . . .	80
5.10	Search menu . . . . .	81
5.11	Option menu . . . . .	81
5.12	Option menu . . . . .	81
5.13	Interface for displaying the search result . . . . .	82

# List of Tables

3.1	Summary of Affordances and Feedback Mechanisms . . . . .	38
4.1	Possible response for a speech act . . . . .	57

# Chapter 1

## Introduction

A user's networked computer is a portal to the digital library of the future. An important component of this is access to distributed heterogeneous services, which mainly include search services and presentation services. Advances in computer power and network connectivity have currently enabled user access to many more distributed services than in the past. This thesis addresses the problem of providing users with access to the above mentioned services of a digital library.

The ideas in this thesis formed a basis for a prototype system. This prototype system has been used to explore issues in interface design, including interaction and architectural issues. This chapter describes the problem and current approaches to it, the context in which this work was carried out, and the groundwork for analysis of user needs that motivates the thesis work. It concludes by summarising the contents of the thesis.

### 1.1 Problem Statement

There is a wide and ever-growing gulf between the distributed information that is available over the Internet and the users who access them. The gulf is wide enough that novice users have trouble crossing it, and the gap is growing wider as more information becomes

available. As more novice users begin to use the internet, two major problems arise: one is specifying all the requirements of the user, and second is finding the relevant information.

Users often have trouble in figuring out how to specify the requirements for a search. This situation is analogous to that of early PC applications wherein; each designer used her own judgement about how controls should look and where they should be placed. The Xerox Star was the first system to provide a uniform Graphical user interface (GUI) environment for multiple applications [Joh89]. The heavily-promoted Macintosh user interface guidelines caused the same sort of standardisation across multiple suppliers of application [Inc87], and this approach is now prevalent with its adoption in Windows95 and Windows98. The work presented here is a first step towards a uniform user interface for varieties of document types such as text and image.

It is likely that the growth of available online information will continue and that a growing proportion of existing information in traditional libraries will become available electronically. This leads to the second problem of finding relevant information, because most searches on World-Wide Web (WWW) often provide a large number of hits and thus overloading the user with information. So the biggest challenge on the web will be to find something that meets the user's specific needs. One solution to overcome this problem is using relevance feedback. Relevance feedback is a mechanism developed in information retrieval area whereby the user can modify an existing query based on available relevant judgements for previously retrieved documents. The goal of relevance feedback is to retrieve and rank highly those documents that are similar to the document(s) the user found to be relevant.

In this thesis, we have proposed one more approach to the above problem of information overloading using agent technology. In this approach we suggest using techniques of co-operation among a team of users or their agents. Besides, the proposed system supports learning and working in a team. By learning and working we mean:



- learning techniques and activities specific to a particular system
- learning issues related to the domain being searched
- sharing hints and tips
- asking for advice on a particular problem

## 1.2 Context: Digital Library

This thesis work was carried out as part of a joint Digital Library project. The database used in this work is the Traquair photographic archive in the Canadian Architecture Collection at McGill University, Montreal. It consists of 7,922 images of architectural buildings and 640 images of silver artifacts. In addition to photographs taken by Ramsay Traquair himself, the archive also includes photographs taken by Edgar Garipy, Livernois Ltd., M.E. Massicotte, Gordon Antoine Neilson, and the Notman Company. Historic architecture in Canada, particularly in the province of Quebec, is documented and so is Quebecs' silver artifacts used for religious and domestic purposes.

As a result of this thesis the collection of data mentioned above is made available to end users for querying and retrieval. The architecture of the prototype is show in Figure 1.1. It has independently developed Z39.50 [Moe95] servers and a separate Z39.50 client, which is a HTTP/Z39.50 gateway client. It provides the users with a uniform interface to the databases and holds detailed information about available servers.

The goal of the prototype is to support search and retrieval between multiple servers and clients. The implementation used Z39.50 to search and retrieve textual data from servers, and HTTP to transfer images between the servers and clients. This process involves handling multiple images associated with the architectural buildings and silver artifacts. An architectural image is described by an object record. One or more digitised images are associated with this object.

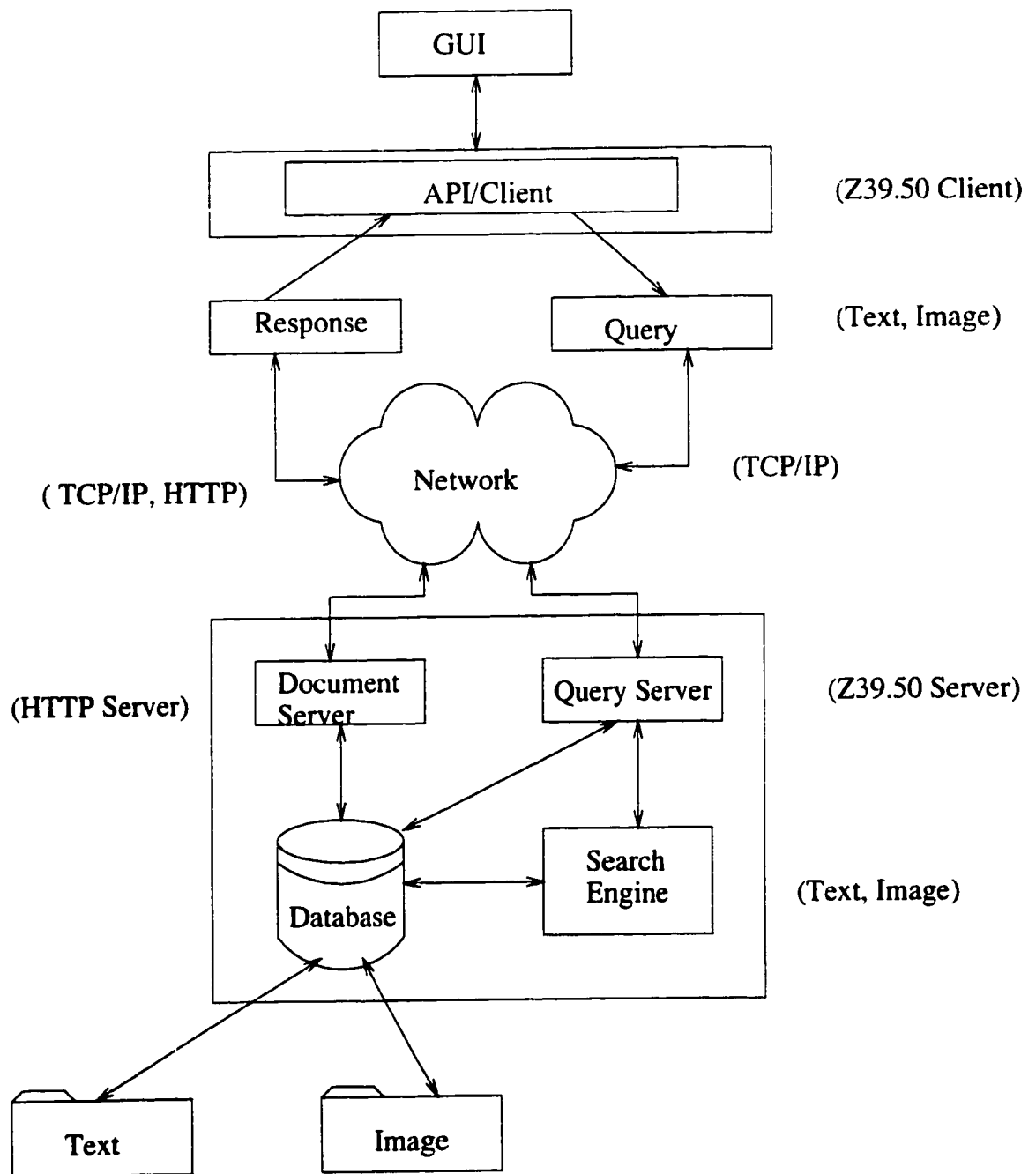


Figure 1.1: System Architecture (TCP/IP: Transmission Control Protocol/Internet Protocol; API: Application programming interface)

## 1.3 Need Analysis

Much of the motivation for this work comes from the library world. Since our vision of the digital library is broad, the prototype described in the previous section is appropriate for tasks beyond the scope of the traditional library.

Digital libraries are more than on-line version of traditional libraries. As Huser put it [HS95] “One of the main goals in developing digital libraries is to provide users with opportunities for accessing and using information in highly flexible and user-oriented ways not available in current information repositories”.

### 1.3.1 Motivation

There is substantial literature related to information retrieval from libraries and on the question of how people access such information. A typical user of a library engages in one or more of the following:

- **Build information based on search results:** People use libraries as tools to accomplish other tasks. These tasks often involve producing “information compounds” such as memos or reports as the end results. Hence, library search tasks can be viewed as producing intermediate results towards this larger goal.
- **Utilise different types of search:** People in library use different types of search. For example they can do Author search, Journal title search, subject search or call number search. In this, the users knows what they are looking for, specify it, and find it.
- **Learn from search results and refine search parameters:** Search in libraries is an iterative process. Reference librarians are trained in library science and they assist in the formation of queries using subject headings, citation, keywords etc. When

computers are used for on-line search, the process can be made interactive. This kind of progressive search is different from the refinement of a query in a search session, which is intended to gradually bring the result set closer to match the user's information needs [Bat89].

- **Save intermediate result sets:** Reference librarians are trained to formulate queries, which are used to find subject headings, which then lead to shelves full of related materials. Subsequently, these materials can be used to find new subject headings, leading to next areas of the shelves.
- **Utilise many information-processing services:** Digital library services range from search and retrieval, to that which help us understand what we have found, to mechanisms that helps manage our results and share them with others.
- **Collaboration:** The use of library resources is often stereotyped as a solitary activity. In addition, there is hardly any mention in the library science and information retrieval literature about the social aspects of information systems. Other than the traditional reference interview, which has been extensively studied, there seems to have been very little consideration of issues of collaboration with other users of the library. We believe that introducing support for collaboration into the process of information retrieval could possibly benefit the user more.

### 1.3.2 Goals

A digital library interface should support the following:

- **User tasks:** The use of library is part of a larger task context. Library users have goals they want to achieve, and individual library activities are important as a means of achieving those goals. The results of an individual search are combined with other searches and the use of other services to achieve the final goal.

- **Sharing and reusing information gathered:** The interface needs to support sharing and reusing information processing knowledge. In a computerised environment, the searches of users can be easily recorded and re-used. The following are the mechanisms for using this information to improve future searching efficiency:

- collaborative filtering [GT94]
- social filtering/recommending [HF95]

The common feature of the above approaches is that one user's activity can benefit another users' search. Hence, it is important to have a well designed interface, so those individuals in a group could share expertise in a simple and effective manner.

## **1.4 Contribution of this thesis**

The wide acceptance of digital library services will depend, to a large extent, on the ease-of-use and ease-of-learning of the user interface. Hence, our goal is to facilitate:

- development of a versatile and user friendly interface for individually performed search, and
- enabling cooperation for sharing and reusing information in a group.

The rest of the thesis is organised as follows:

- Chapter 2 gives a detailed account of agent's classification providing two examples of agents in a digital library.
- Chapter 3 deals with the design of the User Interface for individually performed search.
- Chapter 4 explains in detail a system for cooperative navigation to share and to reuse information gathered from a digital library.

- Chapter 5 describes the implementation of a prototype.
- Chapter 6 provides a conclusion and suggestion for future research.

## Chapter 2

# Literature Survey

In this chapter, we introduce agents and provide a broad overview of various agent-based systems. In particular, the core of this chapter is focussed on the application of agent technology in Digital Libraries.

### 2.1 What is an Agent

By “agent”, we mean a software entity that acts on behalf of a human user. The list of characteristics that have been proposed as desirable qualities an agent should process are as follows [WJ98]:

- **Autonomous:** an agent is able to take the initiative and exercise a non-trivial degree of control over its own actions:
  - **Goal-oriented:** an agent accepts high-level requests indicating what a human wants. Subsequently, it is responsible for deciding how and where to satisfy the requests.
  - **Collaborative:** an agent does not blindly obey commands, but has the ability to modify requests, ask clarification questions, or even refuse to satisfy certain

requests when necessary.

- Flexible: the agent's actions are not "scripted". Further, it can dynamically choose which actions to invoke, and in what sequence, in response to the state of its external environment.
- Self-starting: unlike standard programs which are directly invoked by the user, an agent can sense changes to its environment and decide when to act.
- Temporal continuity: an agent is a continuously running process and not a "one-shot" computation that maps the given input to an output and then terminates.
- Character: an agent has a well-defined, believable "personality" and an internal state.
- Communicative: the agent is able to engage in a complex communication with other agents, including people, in order to obtain information or enlist their help in accomplishing its goals.
- Adaptive: the agent automatically customises itself to the preferences of its user based on previous experience. The agent also automatically adapts to changes in its environment.
- Mobile: an agent is able to transport itself from one machine to another and across different system architectures and platforms.

## 2.2 Classification of agents

A broad classification of agents appears in Nwana [Hya96], where agents are categorised into the following types: collaborative, interface, mobile, information, reactive, hybrid, heterogeneous, and smart.

This section reviews different types of agents with examples which can be used in information retrieval process.



### 2.2.1 Collaborative agents

Collaborative agents emphasise autonomy and cooperation with other agents in order to perform task for their owners. Additionally to have a coordinated set, the agents negotiate to reach mutually acceptable agreements on some matters. The general characteristics of these agents include autonomy, social ability, responsiveness and pro-activeness.

The goal of having a set collaborative agents is to enhance the functionality of the set.

The motivation for collaborative agent systems may include the following:

- To solve problems that are too large for a centralised single agent to accomplish due to resource limitations or the risk of having one centralised system.
- To provide solutions where the expertise is distributed.
- To enhance modularity (which reduces complexity), speed (due to parallelism), reliability (due to redundancy), flexibility (i.e. new tasks are composed more easily from the more modular organisation) and reusability at the knowledge level (hence share-ability of resources).

Examples: The Pleiades project at Carnegie Mellon University(CMU) investigate methods for automated negotiation among collaborative agents, in order to improve their robustness, effectiveness, scalability and maintainability [urla]. The project applies collaborative agents in the domain of Organisational Decision Making over the “InfoSphere”.

### 2.2.2 Interface Agents

Interface agents like collaborative agents also emphasise autonomy and learning in order to perform tasks for their owners. Pattie Maes [Mae], a key proponent of this class of agents, points out that the key metaphor underlying interface agents is that of a *personal assistant*

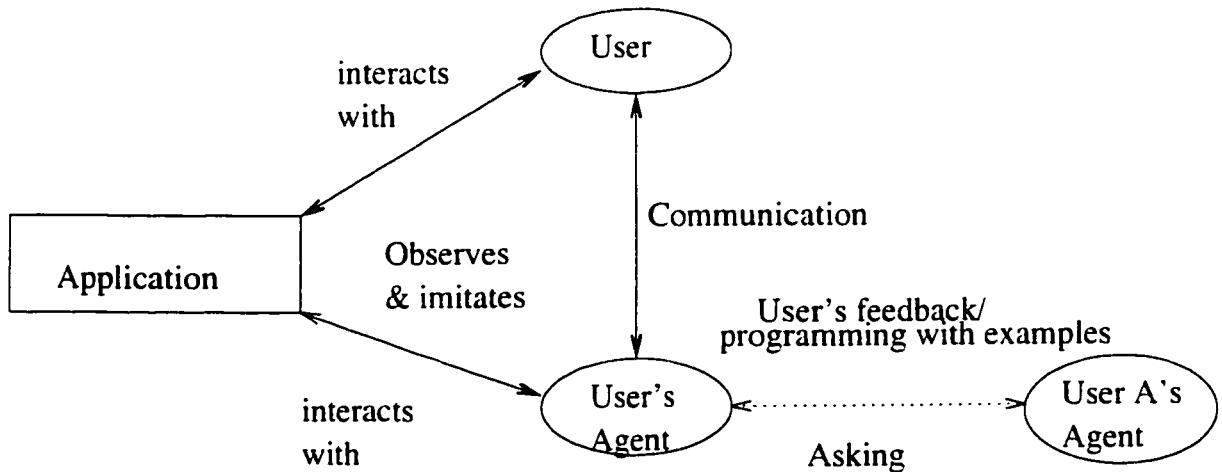


Figure 2.1: Functioning of Interface Agents.

who is *collaborating* with the user in the same work environment. Note the subtle emphasis and distinction between collaborating with the user and collaborating with other agents as is the case with collaborative agents. Collaborating with a user may not require an explicit agent communication language as is required when collaborating with other agents.

Figure 2.1 depicts the functioning of interface agents. Essentially, interface agents support and provide assistance, typically to a user learning to use a particular application such as a spreadsheet or an operating system. The user's agent observes and monitors the actions taken by the user in the interface (i.e. 'watches over the shoulder of its user'), learns new 'short-cuts', and suggests better ways of doing the task. Thus, the user's agent acts as an autonomous personal assistant which cooperates with the user in accomplishing a given task in the application.

The goal is to migrate from the direct manipulation metaphor to one that delegates some of the tasks to (proactive and helpful) a software interface agents in order to accommodate novice users.

The motivation for using interface agents is to eliminate the tedium of humans perform in repetitive action environments, thereby improving productivity and reducing user workload.

Examples:

- Kozierok and Maes [KM93] describe an interface agent called Calendar Agent. This is used for scheduling meetings which is attachable to any application provided it is scriptable and recordable, e.g. scheduling software package. The Calendar Agent assists (i.e. its role is in assisting) its user in scheduling meetings which involves accepting, rejecting, scheduling, negotiating and rescheduling meeting times.
- Liebermann [Lie95] describes an agent called Letizia, whose role is that of a guide assisting web browsing (a keyword and heuristic-based search agent). When users operate their browser, e.g. Netscape, they must state their interests explicitly when using traditional search engines such as Webcrawler or Lycos. The user remains idle while the search is in progress, and likewise, the search engine is idle while the user is browsing the interface. Contrary to the above, Letizia essentially provides a cooperative search between itself and the user.

### 2.2.3 Mobile Agents

Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the world wide web (WWW). Besides they can interact with foreign hosts, gathering information on behalf of its owner and coming 'back home' having performed the duties set by its user. These duties may range from a flight reservation to managing a telecommunications network. These agents can move from one machine to another for task execution, as opposed to static agents, that require information be sent to the machine they reside on.

The motivation for using mobile agents include the following anticipated benefits.

- **Reduced communication costs:** It obviates the need for costly network connections between remote computers, a requirement in remote procedure calls (RPC), thereby

providing a much cheaper alternative.

- Easier coordination: By using mobile agents it is simpler to coordinate a number of remote and independent requests and only collate all the results locally.
- Asynchronous computing: Users can ‘set off’ their mobile agents and do something else and the results will be back in their mailbox, say, at a later time. The agent can operate when the users are not even connected.

Example: The first commercial application was Sony’s Magic Link PDA or personal intelligent communicator (PIC) [urlb]. Essentially, it assists in managing a user’s e-mail, fax, phone and pager as well as linking the user to Telescript-enabled messaging and communication services such as America Online.

#### **2.2.4 Information Agents**

Information Agents also known as Internet agents have been developed because of the demand for tools to manage the rapid growth of information currently being experienced. Information agents perform the role of managing, manipulating or collating information from many distributed sources. Interface or collaborative agents started out quite distinct, but with the spread of the WWW and because of their applicability to this vast WAN, there is now a significant degree of overlap among information, interface and collaborative agents.

The motivation for developing information/internet agents is twofold. Firstly, there is the need for tools to manage the rapid growth of information. Secondly, there are vast financial benefits to be gained. Whoever builds a proactive, dynamic, adaptive and cooperative agent-based WWW information manager is certain to reap enormous financial rewards.

Example: Etzioni and Weld [Eic94] describe a agent called the internet softbot (software robot). It is a fully implemented agent which allows a user to make a high-level request,

and the softbot is able to use search and inference knowledge to determine how to satisfy the request in the internet. In doing so, it is able to tolerate ambiguity, omissions and any errors in the user's request.

## 2.3 Agent Modeling

This section summarises three approaches to agent modeling namely, theory, architectures, and languages.

### 2.3.1 Agent Theory Approaches

**Theory:** An agent is viewed as an entity with knowledge, beliefs, desires and intentions. More generally, they are viewed as intentional systems. Two commonly used attitudes for representing agents are *information attitude* such as belief and knowledge and *proactive-attitude* such as desire, intention, commitment and choice. Currently is unclear which combination of attitudes is appropriate to represent agents. Different researcher's take different approaches to circumvent the above problem.

**Possible Worlds Model by Hintikka:** The possible worlds model for logics of knowledge and belief was originally proposed by Hintikka [Hin62], and subsequently formulated in a normal modal logic using techniques developed by Kripke [Kri63]. Hintikka's insight was to see that an agent's beliefs could be characterized as a set of possible worlds. In this approach, temporal logic was used extensively to model the action.

**Belief and Awareness by Levesque:** Levesque [Lev84] proposed a solution to the problem of making distinction between explicit and implicit belief. An agent has a set of explicit beliefs, and much larger set of implicit beliefs, which are logical consequences of explicit beliefs. A drawback to this solution is that it lacks quantification, and makes incorrect

reasoning at times [Lev84]. To rectify this, Fagin and Halpern[FH85] developed a logic of general awareness.

**Intention by Cohen and Levesque:** A logic developed by Cohen and Levesque (1990) analyses conflict and cooperation in a multi-agent dialogue. A rational balance must be achieved between goals and beliefs of agents. The authors first identified seven properties for agent systems to be satisfied by a theory of intention. Based on this they developed a logic of rational agency. They also introduced several operators in addition to the standard modal operations, and show which of the seven criteria could be satisfied by their theory.

**Rao and Georgeff:** In the work of Cohen & Levesque [CL90], mentioned above, two basic attributes namely, beliefs and goals were used. Further attributes, such as intention, were defined in terms of these. In a related work, Rao and Georgeff [RG91] have developed a logic framework based on Belief, Desire, and Intention (called BDI). Their formalism is based on branching temporal logic which examines how an agent's belief about future affects its desires and intentions.

### 2.3.2 Architectures

**Intelligent Resource Bounded Machine Architecture (IRMA)** was developed as an agent architecture based on the attributes belief, desires, and intentions [BP88]. This architecture has four key symbolic data structures namely, a plan library, and an explicit representations of beliefs, desires, and intentions. Additionally, the architecture has a reasoner, for reasoning about the world; a means-ends analyser, for determining which plans might be used to achieve the agent's intentions; an opportunity analyser, which monitors the environment in order to determine further options for the agent; a filtering process; and a deliberation process. The IRMA architecture has been evaluated in an experimental scenario known as the Tileworld [PR90].

**HOMER** An experiment in the design of intelligent agents was conducted by Vere and Bickmore [VB90]. They argued that the enabling technologies for intelligent agents were sufficiently developed to be able to construct a prototype autonomous agent, with linguistic ability, planning and acting capabilities. They developed such an agent, and named it HOMER. This agent was a simulated robot submarine, which could plan to achieve its instructions, and execute them with appropriate modifications at run time as required.

### 2.3.3 Communication

Formalisms for representing communication in agent theory have tended to be based on speech act theory, as originated by Austin [Aus62], and further developed by Searle [Sea69] and others ([CL90]). Most of the work in speech act theory has been devoted to classifying the various types of speech acts. The two most widely recognised categories of speech acts are *representatives* and *directives*.

**Agent communication languages** [GK94] are based on work in speech act. The best known work on agent communication languages is that by ARPA knowledge sharing effort [Pa92]. This work has been largely devoted to developing two related languages, the knowledge query and the manipulation language (KQML) and the knowledge interchange format (KIF). KQML provides the agent designer with a stranded syntax for messages, and a number of performatives, which described the purpose of the communication, that define the force of a message. On the other-hand KIF provides a syntax for message content. KIF is essentially the first-order predicate calculus, recast in LISP-like syntax.

## 2.4 Application of agent Technology in Digital libraries

Present libraries have the capability to automate tasks such as cataloguing, searching and circulation. These libraries can be seen as the concatenation of an on-line public access

catalogue plus the collection of printed books. On the other hand, it can be argued that digital libraries should be more than simple digital repositories and computerised search engines. Digital libraries should be regarded as dynamic and active providers of what is likely to be rapidly changing information. Therefore, the conventional metaphor for information retrieval is inadequate. Hence, it should be supplemented with an information retrieval technique for dynamically changing information stored in digital libraries. Other important technical challenges which would affect the traditional information retrieval process arise from the need for distributed storage, distributed retrieval and multimedia retrieval.

The following sections give an overview of the various research components of University of Michigan Digital Library (UMDL)[DM96] and on-going research at the University of Sunderland [SB96].

### **2.4.1 University of Michigan Digital Library (UMDL)**

UMDL is organized as a distributed system where the tasks are distributed to numerous specialized, fine-grained modules called agents. This promotes modularity, flexibility and incrementality. These agents have local knowledge about specific tasks and their autonomy. Limiting the complexity of an individual agent simplifies control, promotes reusability, and provides a framework for tackling interoperability problems. Each agent performs a highly specialized library task and has a generic communication interface.

#### **2.4.1.1 Different type of agents in UMDL**

User interface agents, mediator agents and collection interface agents are the three classes of agents that populate UMDL (Figure 2.2).

**User interface agents (UIAs)** The UIA manages the interface that connects human users to UMDL resources. UIA performs the following functions, with assistance from other agents:



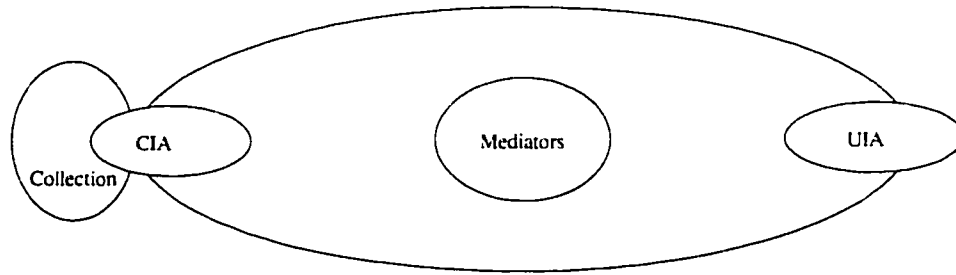


Figure 2.2: UMDL agent types

- express user queries in a form that a search agent can interpret.
- maintain user profiles based on specified, default, and inferred user characteristics.
- customize presentation of query results.
- manages the user's resources available for fee-for service activities.

**Mediator agents** Mediator agents provide intermediate information services. There are different types of mediating agents for finding, processing and delivering information. These agents are distinguished by their specific knowledge and relevant expertise. In UMDL, mediators deal exclusively with other software agents, rather than end users or collections, where collections are bodies of library content. They perform functions such as:

- directing a query from a UIA to a collection.
- monitoring query progress.
- transmitting formats, and
- bookkeeping.

**Collection-interface agents(CIA)** CIA manages the UMDL interface for collections. The collections can comprise of images, structured documents, image collections, and audio and video. In UMDL, queries are submitted to local information repositories to execute the elementary requests on the actual information sources. Each information source is assigned

a dedicated collection of interface agents, which is responsible for maintaining a link between the repository and the rest of the system. These agents are capable of translating query requests, mapping between data types and formats, and resolving schema inconsistencies.

There are two main capabilities of CIA in UMDL. The first capability is to use knowledge supplied by domain specialists about the structure of the documents or other information sources. The second capability is a more dynamic structured activity, based on usage patterns.

The CIA also publishes the contents and capabilities of a collection in the registry.

#### **2.4.1.2 Agent communication in UMDL**

The UMDL tasks require the coordination of multiple specialized agents working together on behalf of users and collection providers. In order to form teams, agent must be able to describe their capabilities to each other in a way that all can understand.

UMDL agents communicate at three distinct levels of abstraction. At the lowest level, agents employ network protocols such as TCP/IP to transmit messages among themselves. Task-specific protocols dictate how the agents interpret and process these messages.

Agents are more likely to be used frequently if they communicate in widely adopted languages. This represents the second level of abstraction. Standards like Z39.50 can be used to provide broader interoperability. This increases the scope of an accessible collection to an agent posing a given query.

A specialized agent's capabilities will remain untapped unless it makes its abilities and location known and participates in team formation. For this propose, special protocols for team formation and negotiation tasks have been defined. These UMDL protocols represent the third level of abstraction in agent communication.

### 2.4.1.3 The Conspectus and the conspectus language

The information in UMDL is enormous. Hence, to limit queries to potentially applicable CIAs, the information space is partitioned into the following two levels [BW94].

- Collection: the set of actual documents.
- Conspectus: includes the content of the collection, search capabilities of the search engine associated with the collection and the structure of the material in the collection.

Thus conspectus is an abstracted description of the aggregate of collections populating the UMDL. The conspectus is written in a language called conspectus language (CL). CL provides a structure for developing compatible representations of collection. Thus, the conspectus provides interoperability for various search and retrieval methods through a common representation over collections. Since conspectus is large in both scope and size, it is distributed and hierarchically organized.

In UMDL, agents are defined by the information content they can deliver, the information services they can render, or both. Agents describe what they can contribute to an agent team and what their limitations are in a conspectus language. Facilitators can also use the C language to describe capabilities required for participation in a team. CL thus serves as a language for both disclosing and querying about language.

Agents communicate using patterns of messages, where the content of the message is specified by CL and sets of “performatives” describing the purpose of the communication. The messages transmitted between the agents describe capabilities, services, and other primitives. For example, all agents use the ASK performative to make requests to the registry for notification about classes of agents with certain capabilities. The registry agent (which is described latter in this section) continues to send information about these agents, as they come on-line, until the UNASK performative is received.

Another example of performative set is TELL, which is typically used in response to an ASK performative. The registry agent uses TELL to send the names of agents that correspond to some capability specification. The registry agent uses the UNTELL performative to express that an agent is no longer available, or that its capabilities have changed.

## **2.4.2 University of Sunderland Digital Library**

University of Sunderland Digital Library [SB96] is based on Hypermedia architecture. This architecture introduce hypermedia agent, i.e. an active atomic data containing component, or more complex hypermedia component which can participate and co-operatively operate in a logically and physically distributed hyper-information environment.

### **2.4.2.1 Agent architecture**

An modular approach is used to define the architecture of hypermedia agents. The hypermedia agent is divided into two parts, an agent head and an agent body (Figure 2.3). The agent head contains all the required information, to enable the hypermedia agents to operate in a co-operative hyper-information environment. The architecture of the agent head is independent of the information source and content. The agent body represents the actual contents of a hypermedia agent, but the contents are always accessed through the agent head. New information can be added to participate in the hyper-information environment and inter-operate with other agents. An agent head is used as a Transducer to mediate between the information source and other agents in the hyper-information environment (Figure 2.4).

Atom hypermedia agents are the actual data containers. Primitive hypermedia agents aggregate related atoms to build a large logical entity. For example, a document can be represented by a primitive agent, which aggregates all atomic agents that constitute it. The hypermedia composite agents supports higher organizational units. The library agent is the

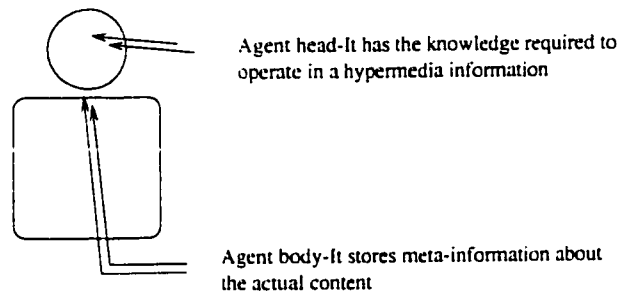


Figure 2.3: Hypermedia Agent Architecture

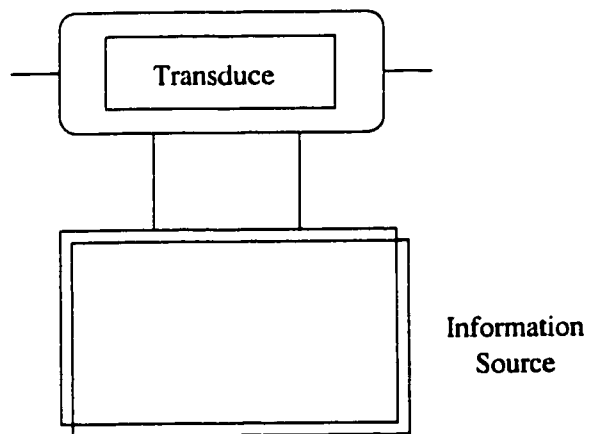


Figure 2.4: A Transducer

highest organizational unit and can be seen as the hypermedia agent representation of a physical library.

#### **2.4.2.2 Agent Communication Language (ACL)**

ACL is the language used in the above model. ACL consists of three parts: its vocabulary, an inner language called KIF (Knowledge Interchange Format), and an outer language called KQML (Knowledge Query and Manipulation Language). KQML can communicate attitudes about information, such as querying, requiring, achieving and offering. KQML is indifferent to the format of the information itself. Thus, KQML messages will contain sub-expressions in other so-called “content Languages”. The vocabulary of ACL is an open-ended dictionary of words which describes a specific application area. Examples of messages expressed in ACL language are listed in the following:

A to B: (ask-if (text, Acropolis)); i.e. Ask B if it is a text agent with information about Acropolis.

A to B: (announce (search, Picaso)); i.e. announce to B the task to “search” for the word Picaso.

B to A: (bid (text, Picaso)); i.e. B offer a bid (e.g. B found documents with the word “Picaso”).

#### **2.4.2.3 Information Retrieval based on Co-operative links**

In digital libraries distribution of the information sources, the heterogeneity of these sources and also the rapidly changing information makes the conventional information retrieval to be reconsidered within the context of digital libraries. A possible solution to the problem of rapidly changing information is the central maintenance of the global index. Distributed information sources can communicate document changes in contents or structure and update

the global index. Unfortunately, in a large distributed information environment this is not cost effective. Another approach to tackle the problem of dynamically changing information is to use dynamic links in order to avoid the need for immediate update. A dynamic link specifies a destination agent in terms of agent specifications. Providing a search based information retrieval mechanism within a hypermedia framework is also a function attached to dynamic link.

Agent-based hypermedia model for digital libraries support co-operative retrieval links as the basic means for search-based distributed information retrieval. Co-operative retrieval links are dynamic links, which when activated, provoke different library agents to be engaged in a co-operative information retrieval process wherein they cooperate, based on negotiation. Co-operative retrieval links, as opposed to static links, are not anchored to a specific location within a document. They are implemented as separate files attached with one or more hypermedia agents. These files should be seen as a set of specifications, in the form of KQML, for searching documents in co-operative library agents.

When a user activates a co-operative link, the search specifications are communicated to the library agents. Here, filters can be used to optimize the selection of library agents from which it is more likely to retrieve the required information. Library agents which receive specifications for searching documents, can apply conventional IR techniques for retrieving documents matching these specifications, or they can alternatively communicate the search specification to their co-operative library agents. Successful hits for documents during this process, are communicated back to the original agent which initiated the co-operative process. Figure 2.5 is a schematic representation of this co-operative information retrieval.

Negotiation between agents is based on the contract-net protocol. Contract-net is a protocol for co-ordinating agents which are able and willing to cooperate. In hypermedia based architecture, the tasks requiring co-operation are information retrieval tasks. Different

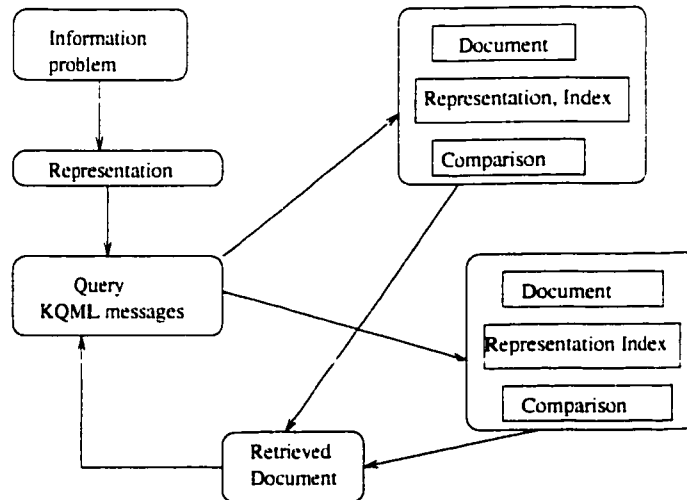


Figure 2.5: A Model for distributed information retrieval

primitive agents may exist which include information with the required specifications. The library agent should announce its intention to allocate the information retrieval task to other primitive agents which are included in its acquaintance list. The primitive agents which receive announcements about the information retrieval task, decide if they can undertake the previously announced responsibility. These agents return bids in order to undertake the responsibility. The departure agent can now decide to which agent it should allocate the responsibility to satisfy the information retrieval task. This decision can be based on previous experience, efficiency, or based directly on the choice of the user.

Co-operative retrieval links in conjunction with the browsing mechanism could provide a rich environment for information retrieval. Information retrieval using co-operative links can be seen as a mechanism which is used to initially locate specific information in large library systems. The user can also refine the specifications of the co-operative links, in order to narrow the number of bids returned by library agents.

The contract-net protocol, like any direct communication protocol could be inefficient in terms of communication cost. Hypermedia model eliminates this problem, since hypermedia agents can be organized as a federation. A federation system (Figure 2.6) is an architecture



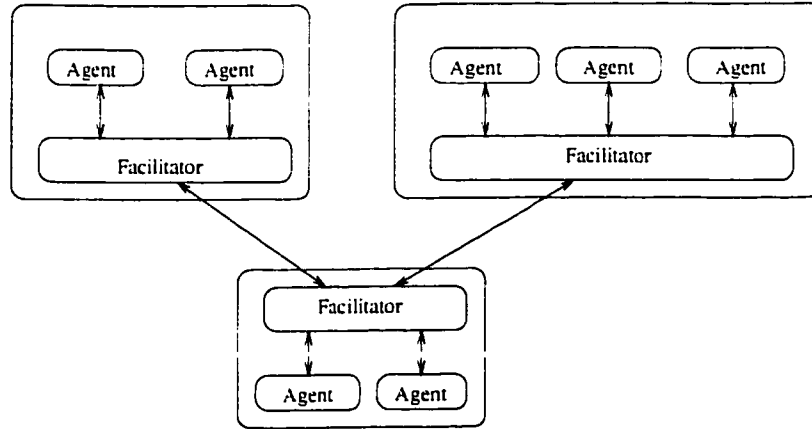


Figure 2.6: Federated System

for indirect communication. Agents do not communicate directly with the other agents, but they communicate through communication facilitators. In hypermedia model, the library agents act as communication facilitators for their “local” hypermedia agents. Using this approach, local documents communicate their document specifications to their local facilitators. Facilitators using this meta-information provided by these local agents, can efficiently communicate with other distributed libraries, for information retrieval.

## 2.5 Summary

This chapter has introduced two sample digital library systems that differ in their respective architectures and designs. The approach followed in our project consists of models, whose origins can be traced back to these systems.

From the two digital libraries described in this chapter, UMDL user interface is closest to our design. Similar to UMDL, our system maintains user profiles based on specified, default, and inferred user characteristics. It uses user profile to customize presentation of query results. In contrast to UMDL, which uses a thesaurus to help the user reformulate his query, our system uses cooperative navigation to help the user in narrowing down the search results. UMDL also supports negotiation and economic transaction by defining protocols for

reaching agreements and specifying the terms of deals between agents. for managing these commitments, and for enforcing and executing the agreements when possible. In contrast, our system uses negotiation by defining protocols for information exchange between agents and for reaching agreements for a proposal.

The digital library from University of Sunderland also uses negotiation for cooperative information retrieval. This negotiation between agents is based on the contract-net protocol. This system is similar to UMDL in terms of functionality, and its most prominent feature is the information retrieval based on cooperative links.

## Chapter 3

# User Interface for Digital Library

The main goal of this thesis is to develop a versatile and user friendly interface for searching a digital library. The proposed user interface system is aimed at “naive” users, wherein it is easy to learn and use the system.

The main functions of the user interface are:

1. expressing user queries in a form that a search agent can interpret (GUI based input is converted to suit the needs of the search agent).
2. filtering the retrieved response set based on the user profile, when necessary.
3. interactive presentation of the retrieved response returned by the search engine in response to the input query.

This chapter focuses on the architecture and visual aspects of the interface. Section 3.1 describes an architecture. Section 3.2 lists a set of requirements for the visual design, Section 3.3 describes the affordances, and shows how they map onto the operational and visual aspects. The final sections provides object class diagram and object interactive diagram of the interface implementation. The design notation used here is UML (Unified Modelling Language).

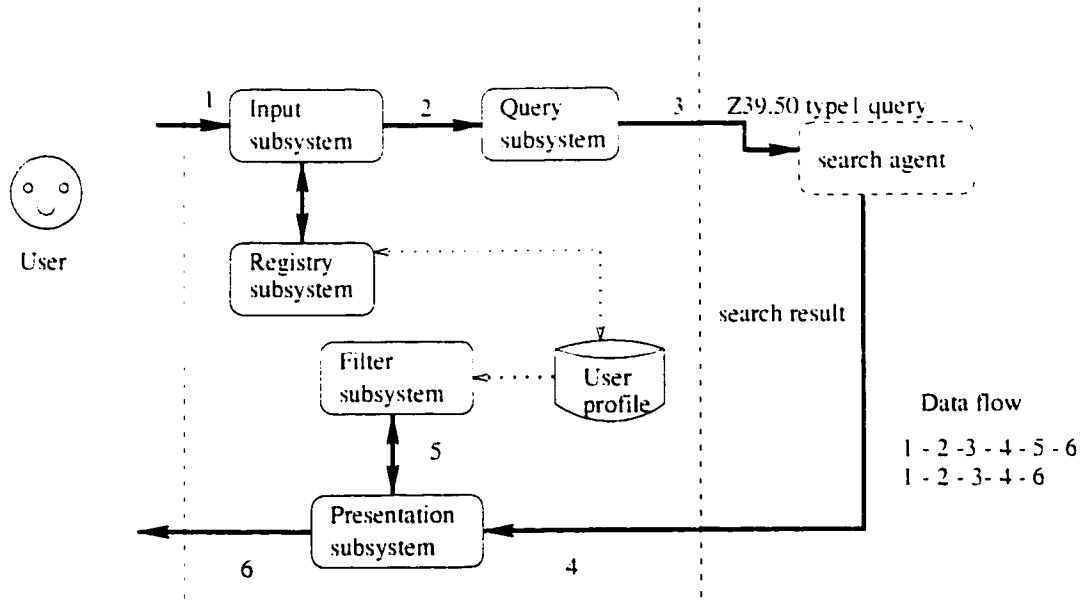


Figure 3.1: System architecture with two alternative data flow sequences

### 3.1 Architecture

The system architecture is composed of five closely integrated subsystems (Figure 3.1). Input subsystem gets input from the user. Registry subsystem gets user information. The Query subsystem processes the user input and generates Reverse Polish Notation(RPN) query, that forms an input for the search agent. Filter subsystem filters the search result depending on the user profile. The user profile permanently stores the profiles of each user known to the system. Finally, the Presentation subsystem displays the search results from the search agent. The subsystems interact with each other as show in the architecture diagram (Figure 3.1). The following sections describes each of the subsystem in detail.

#### 3.1.1 Registry subsystem

To use this application, the user has to register once first. While registering, the system gets information from the user about his preferences, which is subsequently stored in the user

profile. The user can also update the user-profile if his personal information and/or preferences has changed over time. Upon registration the user gets a user name and password, access the digital library.

### 3.1.2 Query subsystem

In the architecture proposed, the Query subsystem receives the query from the Input subsystem. This query consists of two parts. The first part is used by a global search server to determine the sites concerned for querying. It consists of domain name such as art, science or architecture, type of database to search such as text, image or multimedia and database name. The second part of the query contains more specific information namely, phrase(keywords), title, location, photographer's name, image attribute, and type of index used.

Query processing is performed by the Query module, which generate Z39.50 type1 query [Moe95]. Z39.50 is a standard that defines a protocol, which enables remote searching of database and retrieval of the search results. Further, this can search any kind or structure of database.

The Type-1 query is an integral part of Z39.50. This is based on a descriptive technique known as Reverse Polish Notation(RPN), which defines expressions consisting of operators and operands. The structure of type-1 query is as follows:

**RPN-Query ::= Argument|Argument+Argument+Operator**

where

**Argument ::= Operand | RPN-Query**

**Operand ::= AttributeList + Term | ResultSetId | Restriction**

**Restriction ::= ResultSetId + AttributeList,**

**ResultSetId ::= Identification of or pointers to subset of records formed**

**by applying a query.,**

**Term** ::= A word, or a phrase, or a set of words, or a number, or anything that may be searched for in the target database.

**Operator** ::= AND | OR | AND-NOT | Prox

The notation above is used as follows:

::= means 'is defined as'

| means 'or'

+ means 'followed by', where + has precedence over |.

Consider an example query, 'Give me all records in which a title contains any word starting with 'educ'.' . The RPN form of the query is

RPN-Query := AttributeList = (use, title), (relation, equal), (structure, word), (truncation, right), (completeness, incomplete subfield), (position, any), Term = educ

The type-1 query is represented by a tree (Figure 3.2). Each leaf node represents a simple operand. Each non-leaf node represents an operator. For example, consider the query (((PHOTOGRAPHER/Notman or PHOTOGRAPHER/Traquair) and IMAGE/10000) and INDEX/z). The query means that we are looking for a document that contains "Notman" or "Traquair" in the author attribute and the image reference is "10000" and the index method used is "Z". RPN form of the query is : (((PHOTOGRAPHER/Notman PHOTOGRAPHER/Traquair OR) IMAGE/10000 AND)INDEX/z AND).

The query subsystem traverses the tree according to a left post order traversal, to produce a sequence of operands and operators, which is transmitted to the search agent.

### 3.1.3 Filter Subsystem

The presentation subsystem uses filter and User Model to remove less important information from the response set. Filtering is done when the response set is too large to browse sequentially. It is based on the user model and content of the response set.

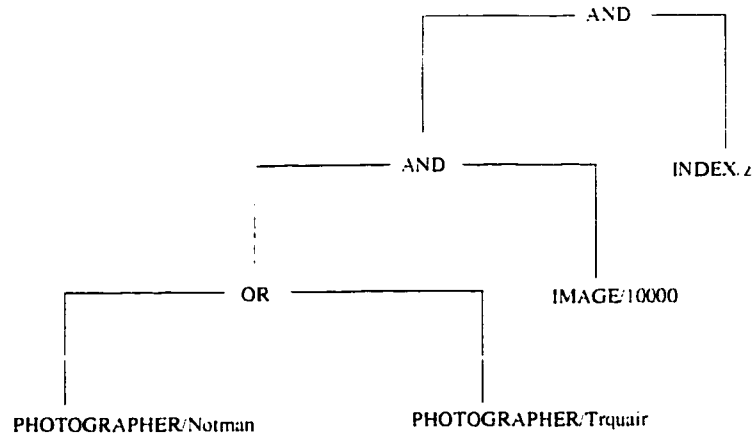


Figure 3.2: RPN query represented by tree

### 3.1.3.1 User Model

The user model is necessary to model the current state of the user and his system preferences. As a result, the user model consists of static entities and is represented by means of a collection of data structures.

The user model consists of two sub-models, namely the task model and a user profile. The task model is a set of tasks and a task may be further be divided into subtasks, recursively. It allows the agent to infer the user's goals at the current level of interaction. This resolves some types of ambiguities without initiating a new dialog with the user. For the digital library application, the tasks and subtasks are represented as follows:

( T1, T2(T21), T3, T4(T41), T5)

where

T1: Receive search results.

T2: Display a random set of search results.

T21: From the displays result select attribute based on which filtering can be done.

T3: Display the search results (after filtering).

T4: Select a search result to follow.

T41: specify the mode (text and/or image).

T5: Store the search result in a file.

The User Profile is organized as a table that can be modified by the user. The table is consulted by the agent whenever filtering of the search results is required. It consists of values for attributes such as photographer, province, building classification and modalities (Text only, image only and text and image). Each of these attributes can consist of single or multiple values.

### **3.1.3.2 Filter**

The search agent locates a set of URLs that matches the user's query and returns the search result (set of URLs) to the presentation agent. The presentation agent in turn displays the URL. The user may select a subset of URLs to view. Whenever the number of URLs exceed a fixed threshold (THETA), the user is queried if he/she would like to retrieve the documents indeed.

The user sets a value for the constant THETA at the beginning of the search. Whenever the number of documents found exceed this threshold value, the user is asked for further action such as cancel, go ahead and retrieve more response or filter the output at current stage and display before proceeding further.

If the user prefers to filter the response before viewing, the system displays a random number of URLs for inspection. By viewing random number of URLs from the response set, the user can get a general idea about using various attribute values to further filter the response set. Once the user selects the attribute by which filtering has to be done, the agent will use the value from the User profile of the user for the corresponding attribute to filter the response set.



## 3.2 Visual Design

The previous section defined an architecture for the prototype without specifying an interaction style. Based on this architecture, it would be possible to build a commandline interface, an audio interface, or a visual interface. Each of these approaches requires additional design decisions, such as identifying the commands or icons.

The actual implementation of the prototype is a visual interface, and this section explores the requirements of the visual design, and then describes the visual design that is implemented in the prototype.

### 3.2.1 Requirements

A large part of visual design is what Liddle [Lid96] calls information display:

*“...information display deals literally with what appears on the screen. It encompasses all those relatively minor aspects, like what window borders and buttons should look like, what fonts are used and where, what icon shapes are used, and so on. This component is important, but is not the crucial concern from the standpoint of usability.”*

Although, information display is a significant component of what we are calling visual design, part of what will be described here is also the control mechanism. In the gap between abstractions and information display are decisions about whether or not to use menus, buttons, dialog boxes, drag and drop, or any of the myriad of other possible graphical user interface techniques.

Many aspects of the visual design of a system are constrained by implementation decisions. If the system is to run on multiple platforms, it may be difficult to get a uniform font. If a standard widget set is used for programming convenience, the designer may not be able to affect the way buttons, text boxes, or menus look. Furthermore, a standard widget set will set up user expectations about how things work, e.g. if the system has a “File” is the

first menu, the user expects “Open” and “Close” to appear on the menu. If the system is to be built with an application builder, the set of interaction mechanisms will be limited to those supported by the tool.

The visual design strongly affects the ease of use of the system. If it clearly communicates the metaphors, and affordances of the system, where affordances is a set of actions suggested to the user by visual representation, new users may be able to begin using the system right away and occasional users will be reminded of what the graphical elements afford. The visual design has its own ontology that interacts with the ontology of the underlying system. A carefully designed visual ontology is part of the design language of a system [Rhe96].

As an example of how implementation decisions, in particular widget set constraints and visual design affect user behaviour, consider this example from the prototype result set component. The example came up in the design of the result set object.

The result set is a composite object that consists of both a collection and a process. It is created by a search service at the time of search, to hold the future results of the search. In addition to inspection, it is possible to ask the result set to get ‘more’ results from the agent. The result set component requires some way to convey to the user the number of results received so far, the number that matched the query of the user, and it needed an affordance with which the user could ask for more results.

The result set has a label on it that conveys the status of the result set. The label is initially set to “searching...” and once the search service finds out the total number of items matched by the query, the label changes to a string indicating the total number of results that match the query. The result set also has a “more” button on it that implements the affordance of asking for more results. The result set has another affordance for inspection and double clicking on the URL set causes the URL to be displayed in detail in the web browser.

### **3.2.2 Visual Design of the Prototype**

The visual design is an important part of any graphical user interface. This work could benefit from the expertise of a professional graphic designer, as much more work in this area is possible. The detailed visual design with figures and screen capture is explained in Chapter 5.

## **3.3 User Interactions**

The set of actions suggested to the user by the visual representation, such as clicking on menu buttons, are the affordances of a system. The means that the system has for informing the user of changes to its internal state are the system's feedback mechanisms.

This prototype implements a unique set of affordances, some of which are similar to affordances in other systems, and some of which make novel variations. The top half of Table 3.1 summarizes the basic affordances, and the bottom half shows various feedback mechanisms. The word 'supports' in the Table 3.1 means, the particular affordance or feedback mechanism is included as a convenience for users who are familiar with other window system,, but not the only way of achieving the function corresponding to that column.

An integral part of the prototype is its use of a web browser as a document viewer and a place to interact with HTML forms. HTML, with its rich text, hypertext links, and forms, is used to give more details of the result than can be expressed in the limited screen space of the user interface. More traditional text manipulations such as selecting words and following hypertext links can be used in the web browser.

The next two sections describe the affordances and feedback mechanisms.

<b>Affordance</b>	<b>Inspection</b>	<b>Editing</b>	<b>Process Initiation</b>	<b>Process Monitoring</b>
Point	yes			
Select			supports	
Activate	yes		supports	yes
Button press			yes	
Menu selection			supports	
Text entry box		yes		
Follow link	yes		supports	yes
Fill in form		yes	supports	
<b>Feedback</b>				
Change text		yes	supports	yes
Send HTML to browser	yes	yes	supports	

Table 3.1: Summary of Affordances and Feedback Mechanisms

### 3.3.1 Affordances

This is not a one - to - one mapping between the affordances provided in the prototype and the requirements described in Section 3.2. The prototype has been designed with following set of assumptions that limit the set of possible affordances:

- The user will interact with the prototype with a pointing device such as a mouse and a text entry device such as a keyboard.
- The prototype will run on multiple platforms, where details like the exact keyboard layout or the number of buttons on the mouse are not known at design time.
- A set of standard GUI widgets is available that the end user can understand, such as text entry boxes and hyperlinks in the web browser.

Most of the affordances of the prototype can be done without the keyboard, using only the pointing device.

#### **3.3.1.1 Pointing**

The most basic affordance in the prototype is pointing. Pointing is a simpler operation than clicking, double - clicking or dragging because the mouse button is not involved. When the user moves the mouse over the element of a prototype object, a small box appears with a short description of the object. Such pop - up message boxes have become a standard feature of application programs and graphical operating system interfaces (they are called balloon help on Macintoshes and tool tips in Microsoft products).

Pointing supports the user's need to inspect the prototype components with progressive disclosure. It allows the elements of components to be drawn without any text, so that hundreds of objects can be drawn in a small amount of screen real estate. The text in the pop - up message box is only the first step in inspecting the object: if more information is required, the user can use help or click on the components.

#### **3.3.1.2 Selection**

Selection is the operation of choosing graphical objects to work on and communicating the choice to the system. Simple selection is done by pointing at a selectable object and clicking (pressing and releasing) the (primary) mouse button in most systems. Single clicking with a modifier key such as the shift - key - down allows multiple objects to be selected.

Some objects in the system with selection are not selectable because they do not respond to mouse clicks, or because they are buttons: objects that do something when clicked. In web browsers, hyperlinks behave as buttons.

#### **3.3.1.3 Activate**

An object is activated when the user indicates that it has to do some default action, typically by double - clicking on it. On the Macintosh, double - click activation is used to open a

document or invoke an application program. Double-clicking was chosen for the Macintosh because singleclicking was reserved for selection and dragging. In our current prototype, activate is invoked by double-clicking or by clicking the right button once.

Although, single-right-click is just a convenient alias for double-click, using it causes confusion when users move back and forth between the prototype interface and the HTML browser, since following a link in the browser is done with single-left-click. This is another example of a tradeoff forced upon the prototype by the choice to make use of existing tools. This problem comes up in other systems as well: Macintosh Performa computers, a line designed for home users, has a feature called the “launcher” that provides a user-configurable set of buttons that launch applications. The launcher uses single-click to start applications, and so requires an extra affordance to allow the buttons to be dragged. In that case, the hidden affordance of holding the option key while dragging was chosen.

The prototype implements activate in a way that is more general than double-clicking on an icon in a standard desktop interface: The component that receives the activate message will pass information to the system about which graphic element was activated (such as the reference element or a text label), so that it can tailor its response to the activation based on a finer granularity.

#### **3.3.1.4 Other affordances**

The prototype makes use of a limited set of available tools for a few of its affordances. It uses the standard button, and text entry box widgets in some components, and it has a menu bar for less common commands. Within the web browser, it makes use of links.

The standard widgets have been a programming convenience and have caused the interface to behave more as users expect. Standard GUI buttons highlight in a way that users expect when the mouse moves over them. The downside of using the widgets is that they cannot easily be dragged since they handle mouse events themselves. They also do not behave the

same way as other elements on the canvas do. HTML links and forms have proven very versatile in the prototype.

### **3.3.2 Feedback mechanisms**

The other important system interaction is feedback. Any system needs mechanisms to convey its state, and to inform the user of problems and progress, and rich ways to respond to user actions. In this prototype, it is done by changing the text of components, and by sending HTML to the web browser. These mechanisms are described in the next two sections.

#### **3.3.2.1 Change text**

On a spectrum from pure textual to pure graphical, the visual design of the prototype has tended toward the pure graphical. The text labels that it does have are a very important mechanism for communicating with the user. Text labels can be changed in response to user events or system events, and provide a convenient way to convey the state of a process to the user. For example, search has a status label that can have the value “searching..” or total number of items found or “not found”.

#### **3.3.2.2 Send HTML to web browser**

The final feedback mechanism in the prototype is sending text to the web browser. This operation supports inspection since HTML pages can contain much more information about a result than the list of URLs. It also supports editing, since HTML pages can contain forms. The prototype send pages to the web browser as a result of an explicit user action.

This mechanism could be used in process initiation i.e., when a service is invoked, it could display information about the invocation, such as warnings in the web browser. In the current implementation this mechanism is not used.

## 3.4 Class Diagrams

The architecture shown in Figure 3.1 is implemented using a set of interacting classes. This section presents the class diagrams, wherein a class diagram describes the types of objects in the system and various kinds of relationships that exist among different classes. The perspective used in presenting the class diagrams is the implementation perspective, which has an impact on the way one should interpret the diagram. The implementation perspective has real classes and one can lay the implementation bare.

Aggregation and inheritance are two commonly used relationships. For example in Figure 3.4, Attribute panel class is an aggregation of various ComboBoxs, label, panel and a textLabel. At the same time, label5 is inherited from label class.

Another relationship used is dependency. Dependency is shown in Figure 3.7 as dotted arrows. A dependence exists between two elements if changes to the definition of one element may cause changes to the other. Dependencies between classes exist for various reasons: One class sends a message to another; one class has another as part of its data; one class mentions another as a parameter to an operation.

Figure 3.3 shows the main class diagram, Figure 3.7 shows the class digram for filtering and presentation of the result and Figure 3.4, 3.5, 3.6 shows class diagram that implements the GUI part of the prototype. The detail description of the classes are provided in Chapter 5.



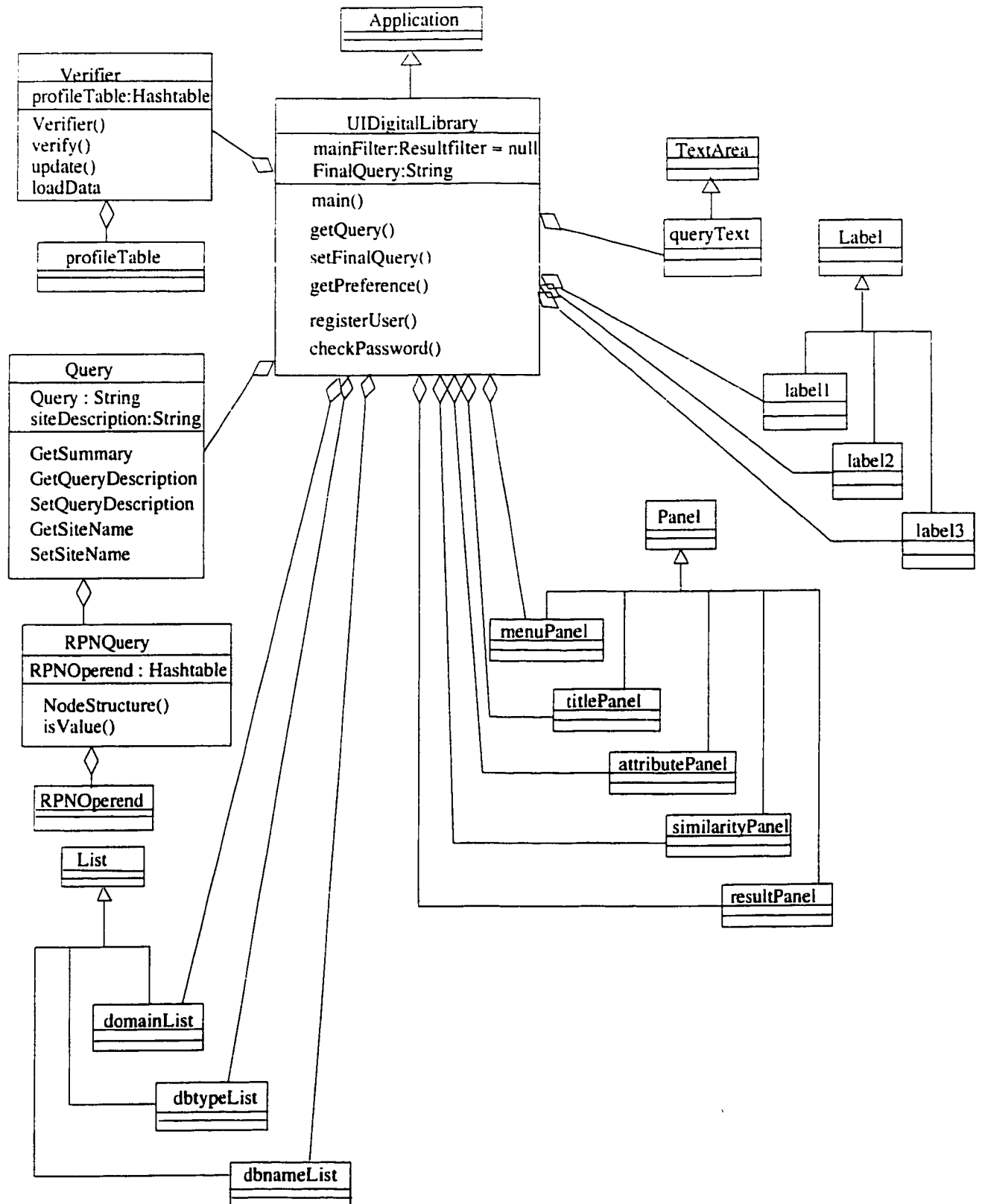


Figure 3.3: Main Class Diagram

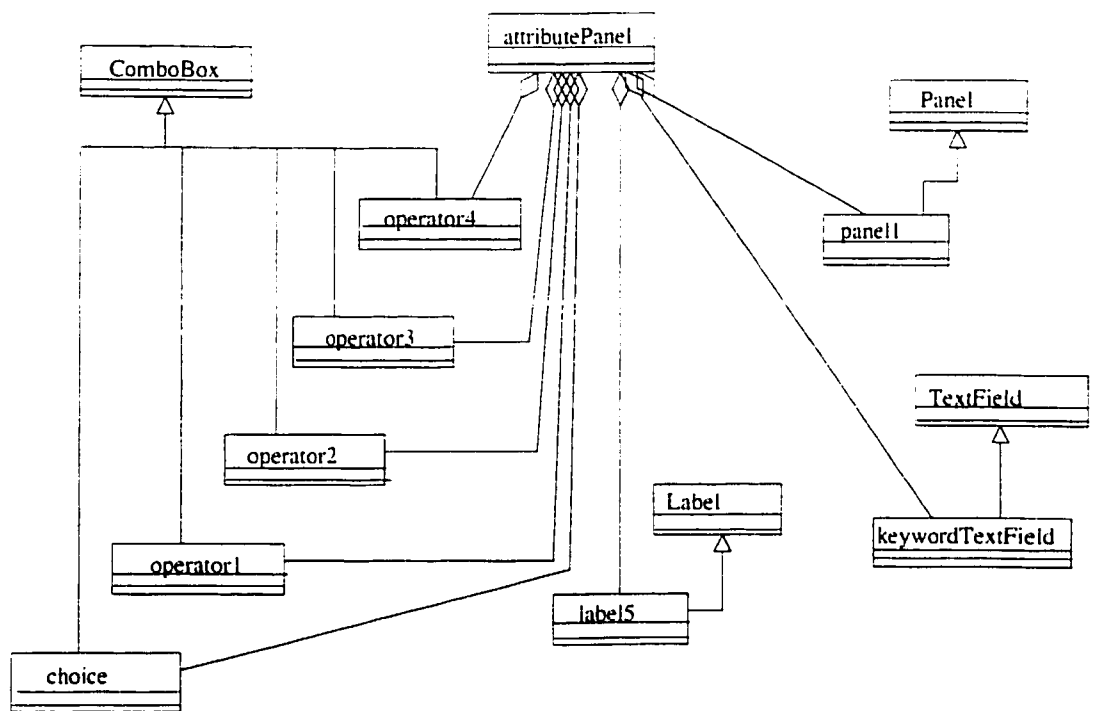


Figure 3.4: Class Diagram for Attribute Panel

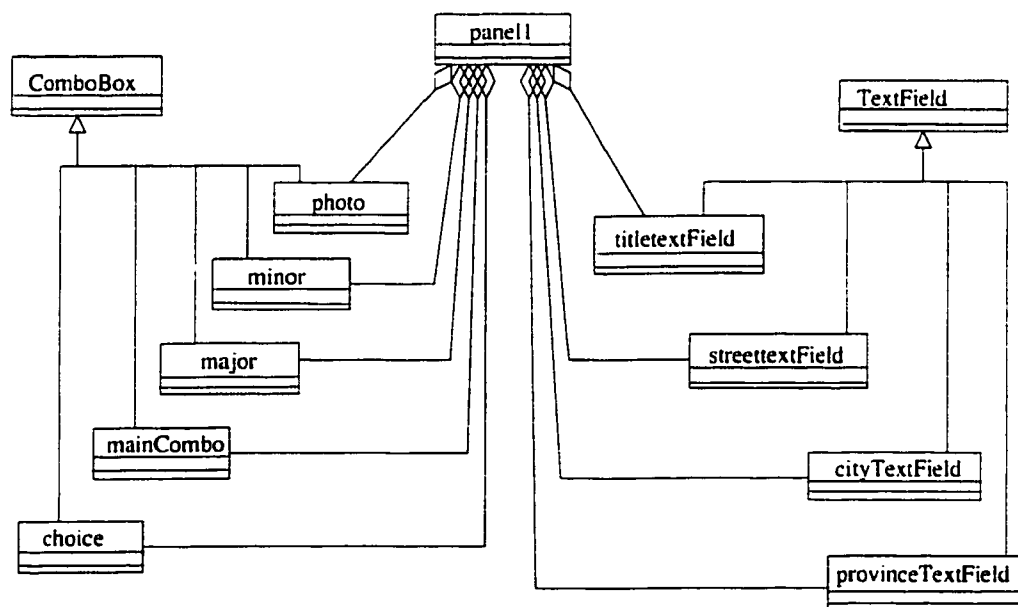


Figure 3.5: Class Diagram for panel1

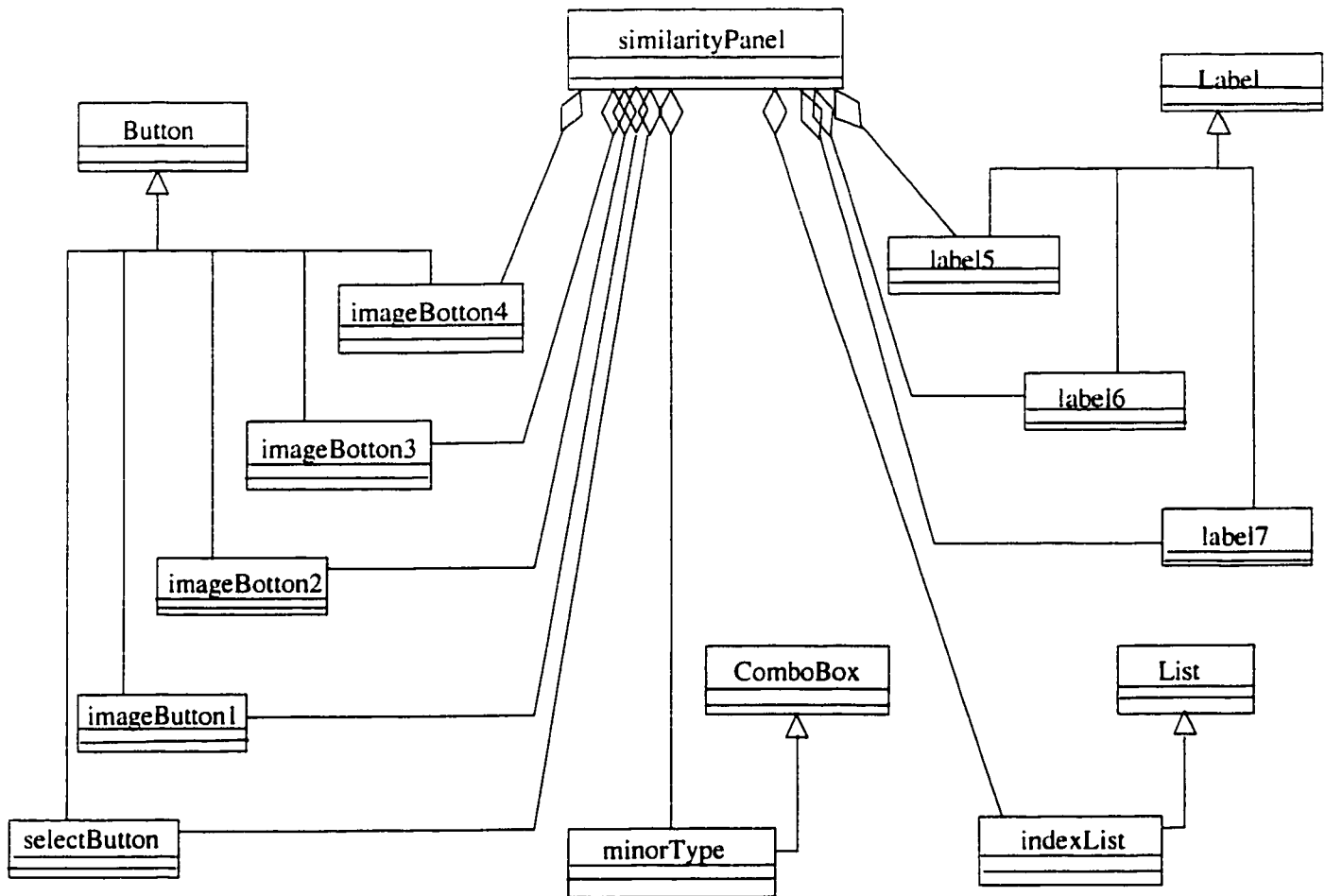


Figure 3.6: Class Diagram for Similarity panel



### 3.5 Object Interaction Diagram

This section includes the object interaction diagrams used in the design of the user interface for a digital library. Interaction diagrams are models that describe how groups of objects collaborate in some behaviour. These diagrams show a number of object examples and the messages that pass between these objects. There are two kinds of interaction diagrams namely, sequence diagrams and collaboration diagrams. We use sequence diagrams to explain the object interaction. Within a sequence diagram, an object is shown as a box at the top of a dashed vertical line ( see Figure 3.8). This vertical line is called the object's lifeline. The lifeline represents the object's life during the interaction.

Each message is represented by an arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labelled at minimum with the message name and arguments. In addition, some control information can also be there. As we can see, Figure 3.8 is very simple and has immediate visual appeal. This is its greatest strength.

One of the hardest things to understand in an object-oriented program is the overall flow of control. A good design has a lot of small methods in different classes, and at times it can be tricky to figure out the overall sequence of behaviour. One can end up looking at the code trying to find the sequence and sequence diagram helps to see this. Figure 3.8 shows the high level interactive diagram for the system. Following this, Figure 3.9 shows the detailed interactions among the objects in the user interface.

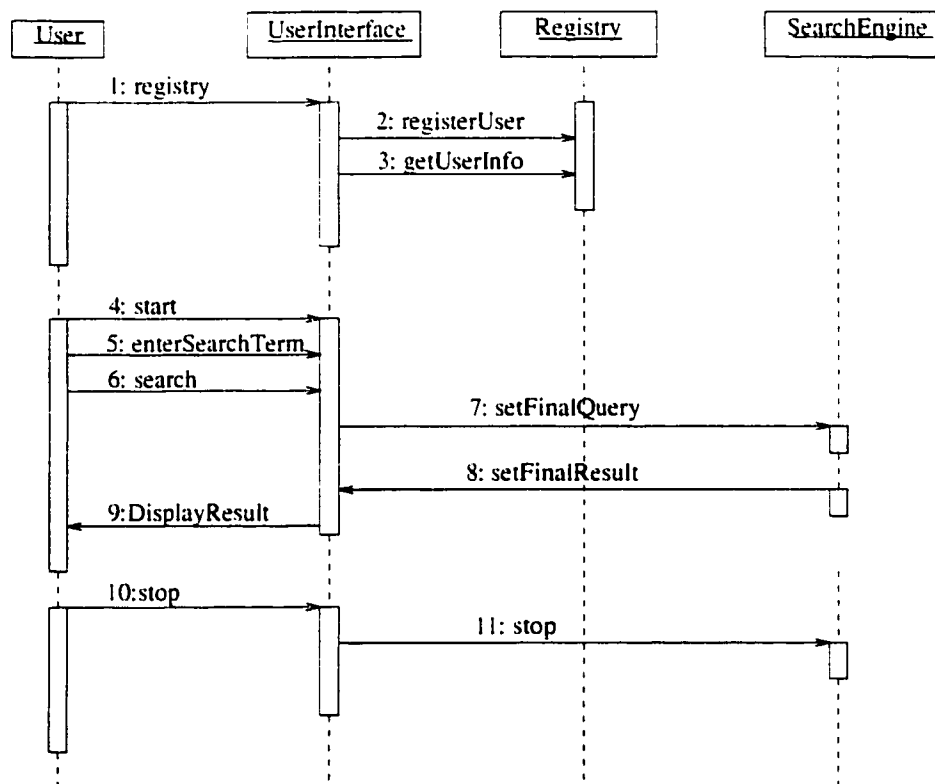


Figure 3.8: Initial object interaction diagram

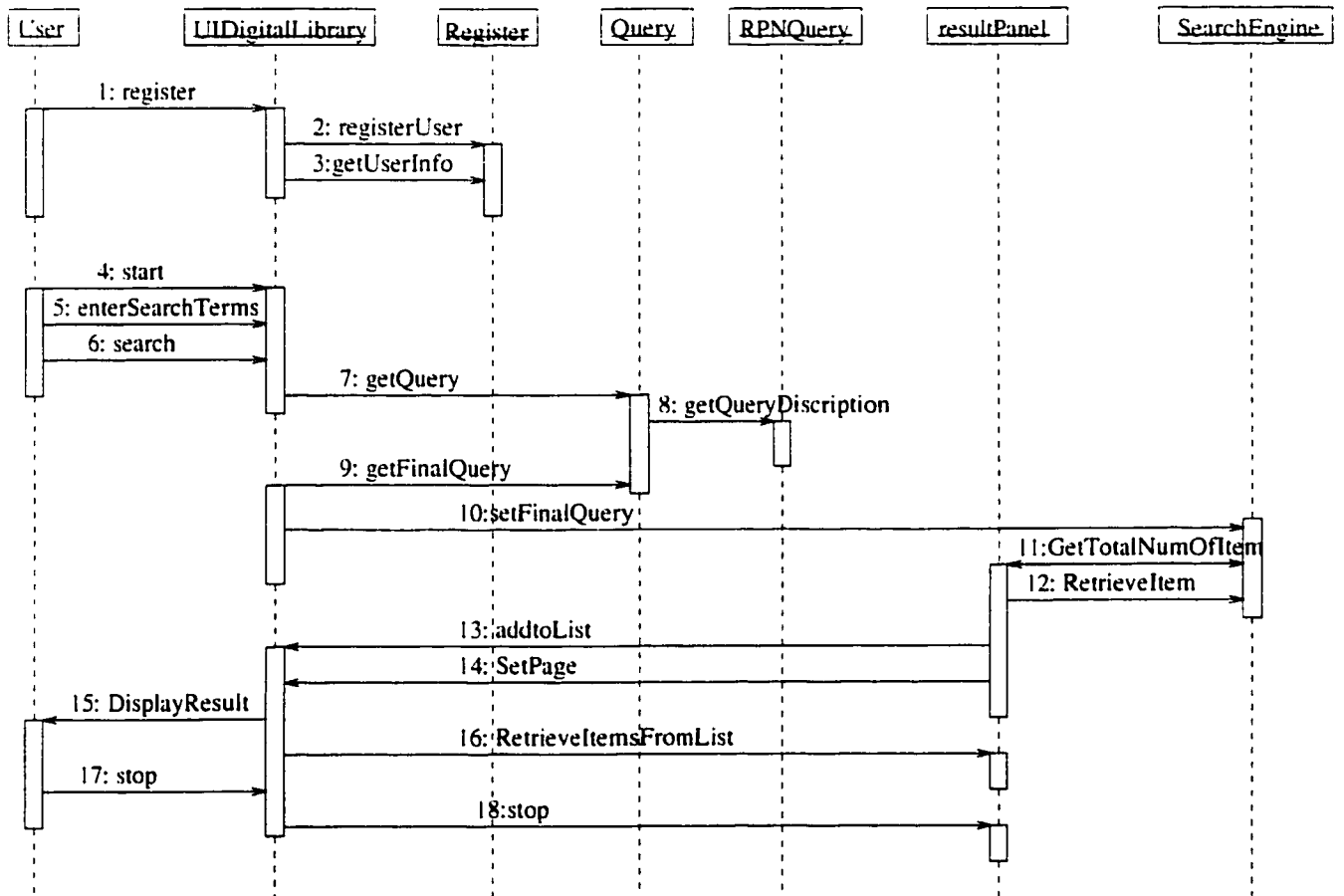


Figure 3.9: Detailed object interaction diagram (when Total number of documents is less than Theta)

## Chapter 4

# Cooperative navigation in digital library

In this chapter, we propose an approach to address the problem of “information overloading” while a user is accessing digital libraries. It makes use of cooperation among a team of users or their agents. Often people work as a team to achieve a large goal. This team of people is herein referred to as a “work group”. Let an agent assist every member of the work group. Cooperation among members of the work group is treated synonymously with the cooperation among their agents. The members of a work group help each other in finding the information relevant to their individual goals or the shared common goal of the team. Apart from retrieving information, some members of a work group might disseminate some relevant information they found to other members.

A cooperative navigation in which one user benefits from another user’s experience is desirable under the following cases:

- When the output from the search engine is too large for the user to find the relevant documents, the agent of one user can cooperate with the agents of others in the work group to find the relevant information.



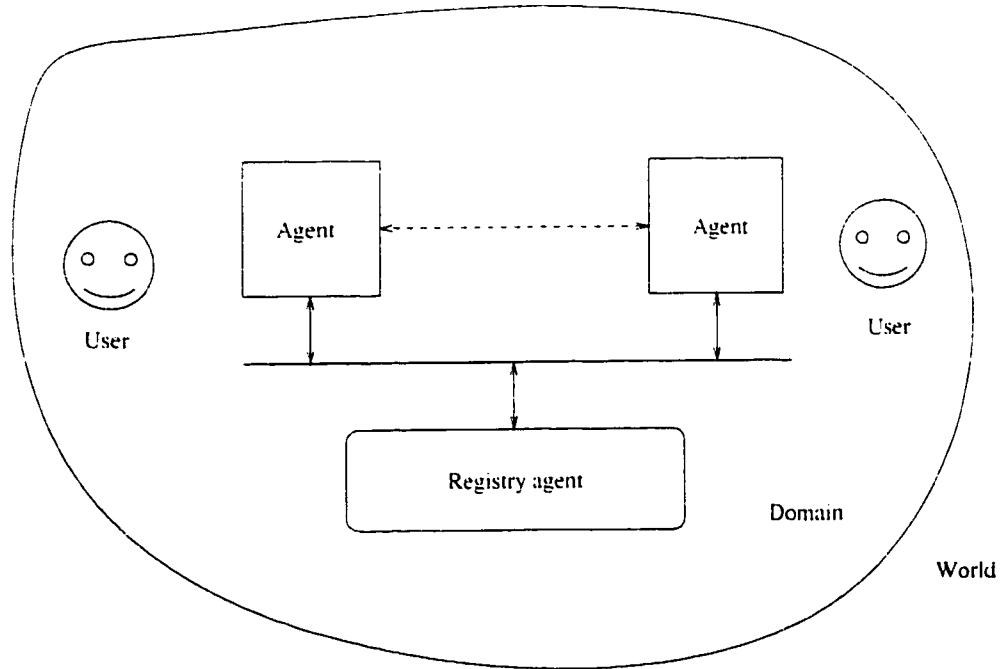


Figure 4.1: System framework for cooperative navigation. *The dotted line shows the communication between agents. All agents communicate with the registry, which is shown by a solid line.*

- Using the knowledge of the members of a work group, the recall and precision could be improved.

The intention of this chapter is to focus and identify a framework that integrates a set of homogeneous agents to perform a cooperative search. In this process, a conflict or disagreement may arise between agents which must be detected and resolved.

## 4.1 System Design

The overall framework of the system (Figure 4.1 ) and its modules (Figure 4.2 ) are described in this section.

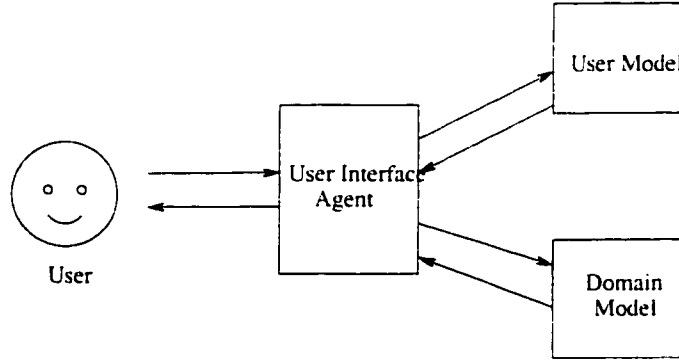


Figure 4.2: System architecture

#### 4.1.1 Agent

An agent is constructed to accomplish a well defined task. For example, an expert system that mimics a reference librarian to help the user to focus his/her search for documents can be viewed as a library agent. Given any two agents in an environment, the following is apparent: either they are responsible for a particular task, or they are responsible for different tasks. Thus, it is possible to partition the set of agents in an environment into two sets, set of homogeneous agents and set of heterogeneous agents.

**Definition 1** (Homogeneous and Heterogeneous Agents) *Two agents  $Agent_1$  and  $Agent_2$  are homogeneous iff they are constructed for solving the same task, otherwise they are said to be heterogeneous. [GRSR97]*

Note that we use task at the level of granularity to distinguish between homogeneous and heterogeneous agents and not at the level of implementation. Thus, two agents for cooperative navigation, say one written in C language and the other written in Java, are treated as homogeneous in this framework. This is intentional, as developers are free to choose an implementation that best suits the environment for which an agent is being constructed. A set of homogeneous agents is capable of communicating with each other. In addition, one should also maintain the list of agents in an environment, the nature of their tasks, etc., in order to facilitate communication between agents and to allow a (new) user to query the

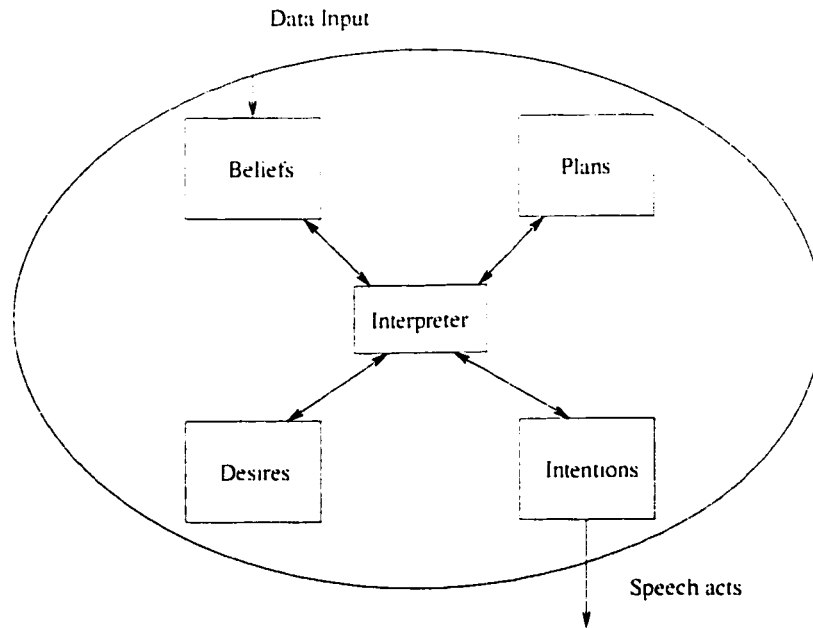


Figure 4.3: BDI architecture

agent services available. These issues bring in a notion of *agent architecture*, *agent registry* and *agent language*.

**Agent Architecture:** The agent architecture is based on BDI (Belief, Desire and Intention) architecture (Figure 4.3), which contains the following four key data structures [GL87]:

1. An agent's *beliefs* correspond to information the agent has about the world.
2. An agent's *desires* correspond to tasks allocated to or undertaken by it.
3. An agent's *intentions* represent desires that it is committed to achieving.
4. The final data structure in a BDI agent is a *plan library*. The plan library has a set of plans. A plan specifies the course of action followed by an agent in order to achieve its intentions.

The interpreter in Figure 4.3 is responsible for updating beliefs based observation made of the world, generating new desires (tasks) on the basis of new beliefs, and selecting from the

set of currently active desires some subset to act as intentions. Finally, the interpreter must select an action to perform on the basis of the agent's current intentions and procedural knowledge.

**Registry Agent:** These agents are first registered with the central registry agent. The registry agent provides information about where a particular agent resides in the physical space. The registry agent contains one entry for each user agent and its associated details such as, the owner of the agent. The list of services provided by the agent is also stored in the registry.

#### 4.1.2 The Agent Language

**Definition 2** (Agent Language) *An agent language  $L$  is a set of syntactic constructs that an agent can parse into acts that are meaningful to the task associated with the agent. An agent that can parse such a language  $L$  is said to speak that language.*

An agent language is also said to define the *interface* of an agent because it is the only means by which other agents can invoke the services of an agent. For example, an agent would parse the construct  $(ask, jaya, G_d)$  to ask jaya (user agent) information about graph  $G_d$ .

Agent Communication Languages are generally based on the Speech act [Sea69] theory. In this, the speaker/hearer model needs to be understood. If one person was to comment to a second person regarding an observation such as “the weather is delightful today” the first person is considered to play the role of a speaker and the second person a hearer. These roles are then reversed for a response from the second person. This pair (inquiry-response) or a sequence of such pairs constitute what is known as “a conversation”.

Unfortunately, there are problems with the work described in the literature that are related to our purposes. These problems can be illustrated by reference to Allen [All87].

For example, Allen defines a number of discourse acts involving actions by both agents in the dialogue as compounds of four primitive speech acts: REQUEST, INFORM, INFORMIF and INFORMREF. Thus the dialogue act ASKIF is defined as a REQUEST that the hearer perform an INFORMIF act, i.e.

(REQUEST  $x\ y$  (INFORMIF  $y\ x\ p$ ))

rather than communicating the speaker's intention. The REQUEST act has the effect that

(WANT $_y$  (INFORMIF  $y\ x\ p$ ))

which is an implicit precondition of

(INFORMIF  $y\ x\ p$ )

which when performed has the effect

(KNOWIF  $x\ y$ )

However, Allen's speech acts make a number of assumptions which are not valid in our context. Because these assumptions form part of the definition of the act itself, they make analysis of the communication difficult. Moreover, Allen's speech acts are intended to serve as a basis for advanced dialogue planning rather than as a basis for the actual detailed conduct of a dialogue.

We therefore define our own speech acts to make as few assumptions as possible about the effects of the act and whether the agents are co-operative etc., and we characterise those additional assumptions that we make as preconditions of the speech act.

We use five speech acts namely, ask, answer, accept, reject and compromise. Each speech act has number of preconditions which must be true for the successful performance of the act.

*(Notations :- I: Intention; B: Belief; s: Speaker; h: Hearer; P: Predicate)*

- **Ask**

(ask, s, h, P): The speaker 's' "asks" the hearer 'h' about the predicate P. The preconditions for the utterance of Ask act are:

1. There exist 'x' such that  $P(x)$  is True.
2. The speaker 's' intends to believe  $P(x)$  after receiving the response from h.
3. Speaker 's' believes that it currently does not know whether  $P(x)$  is True or false but believes that the hearer 'h' knows  $P(x)$ .

- **Answer**

(answer s, h, P): Speaker 's' "answers" about predicate P to hearer 'h'.

The preconditions for the utterance of answer act are:

1. There exist 'x' such that  $P(x)$  is True.
2. The speaker believes  $P(x)$  is True

- **Accept**

(accept s, h, P): The speaker 's' "accepts" the predicate P, about which the hearer 'h' responded.

The precondition for the utterance is that both the speaker 's' and hearer 'h' believe in P.

- **Reject**

(reject s, h, P): The speaker 's' "rejects" the predicate P, about which the hearer 'h' responded.

The precondition for the utterance of reject act is that hearer 'h' believes in P and speaker 's' does not believe in P.

- **Compromise**

(compromise s, h, P): The speaker 's' needs some clarification (in the case of two agents in a work group) or has to compromise by suggesting modification to the predicate P. Then 's' uses "compromise" speech act.

The preconditions for the utterance are:

A's Question to B	B's Responses to A
ask	answer
answer	accept, reject, compromise
compromise	accept, reject, compromise

Table 4.1: Possible response for a speech act

1. There exist 'x' such that  $P(x)$  is true.
2. The speaker 's' believes in  $P(x)$ .
3. Speaker 's' intends hearer 'h' to believe in  $P(x)$ .
4. Speaker 's' believes that hearer 'h' does not believe in  $P(x)$ .

We have identified the possible communicative outcomes for a given speech act and defined how an agent should respond to that speech act in a cooperative environment. For each of the speech acts, we have identified one or more responses (Table 4.1). For example, if the 'act' is a question, the only response might be 'answer'.

### 4.1.3 System Architecture

In general, three essential levels of abstraction should be considered as part of the requirements to building an agent. They are as follows:

- The *domain model*, or a description of the characteristics of the operating domain. For example, an agent whose task is "searching the database" would base itself on the domain model that describes methods of searching, retrieving etc. In contrast, an agent for "organising the search results" would rely upon a totally different model that describes methods for receiving, storing, sorting etc.
- The *user model*, or a description of the characteristics of the user in a domain. The user model is important to an agent for several reasons: (1) to adapt itself to user characteristics. For example, an user agent can mimic user characteristics by sending

different messages to different people when informing the user of new incoming documents that will be of interest to the user. (2) to limit the amount of information that should be output to the user and (3) to determine appropriate temporal (*when* to assist?) and spatial (*how* to assist?) aspects of its response.

- The *world model*, or a description of the set of domains in which the agent is supposed to exist. The world would also contain other (possibly heterogeneous) agents with whom a given agent should co-exist.

The software architecture of an agent is often dictated by its operating domain and the characteristics of the users in the domain. For example, in a domain where users are expected to exhibit same (or similar) repetitive actions, a knowledge-based approach is useful by encoding a set of rules for the system to respond. Such an approach is also useful whenever the nature of domain involves searching based on a fixed set of resource attributes. On the other hand, if the users of a domain exhibit repetitive actions that is different for each user, then a machine learning approach is desirable since the agent would customise itself to a given user profile by learning his/her characteristics over time.

In general, an agent that acts as an intelligent user interface should contain the following components:

1. a knowledge base to apply selective rules based upon the context of the user for his/her assistance,
2. a learning module to learn user preferences and behaviour over time to adapt itself to the user, and
3. ability to learn through exchanges from other (possibly heterogeneous) agents.



## 4.2 Functions of the system

Let us consider a response set that is large enough to warrant navigation by the user. This set can be treated as a sub-graph, if the digital library itself is viewed as a very large graph. At the higher level of abstraction, the nodes correspond to sites (and document items within the site) and the edges connecting two nodes can be viewed as a labelled hyper-link. We call this graph as “site-document” graph. One functionality of the system is to understand how to navigate in this graph, that is how to go from one site to another. Navigation in the site-document graph can be achieved by one agent sharing of information among other agents. This sharing presupposes the cooperation among agents.

The following assumptions are made while defining the functionality of the system:

- Users are willing to provide personal information.
- Search tips and search results from a site are saved and useful information are synthesised.

**Hypothesis 1** *Cooperative search will improve search quality and decrease the time to find relevant information.*

### 4.2.1 Navigation in the site-document graph

Navigation can take place at two levels namely, site level and document level. Site level represents URL or sites in the Internet, while document level represents documents relevant to the query. A Site-Document graph  $G$  (Figure 4.4) is drawn to show the interconnection between site level and document level. In this, navigation function  $S$  is used to determine moves within a level (from one site to another site or from one document to another document).

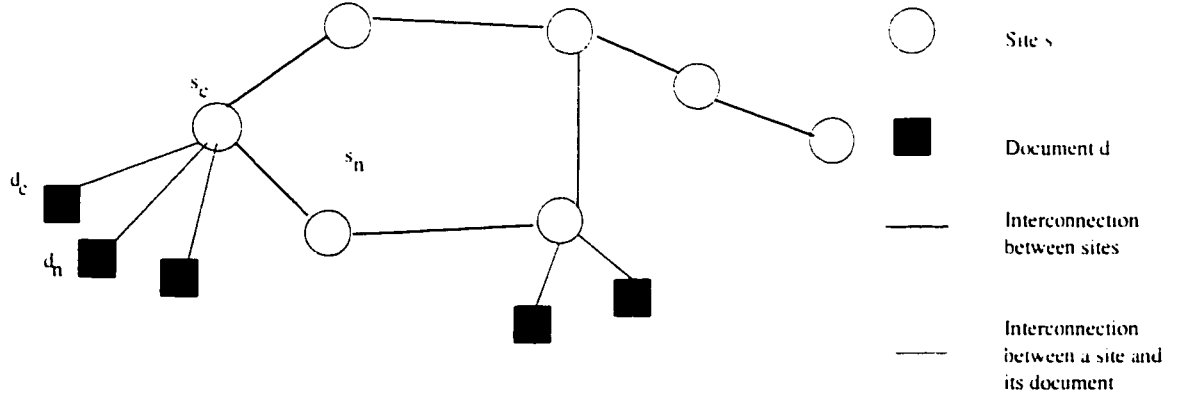


Figure 4.4: Site-Document graph

Given a graph  $G$ , we can determine  $s_n$ , where  $s_n$  is the next site to visit from the current site  $s_c$  by using the function  $S$ .

$$S(G, s_c) = s_n \quad (1)$$

Similarly, we use a document navigation function to navigate at the document level within a given site.

$$D(s_c, d_c) = d_n \quad (2)$$

where  $D$  is document navigation function;  $s_c$  is current site;  $d_c$  is current document;  $d_n$  is next document at the same site.

The Choice function  $C$  decides which level to navigate, site level or document level

$$C(G_m) = S \setminus D \quad (3)$$

where  $G_m$  is a Site-Document graph in which nodes are labelled.  $S$  is site navigation function;  $D$  is document navigation function and choice function selects one of  $S$  or  $D$ .

In a Site-Document marked graph all the sites are marked as '0' initially. A site is marked as '1' if all the documents of that site are visited, and it is marked as ' $\emptyset$ ' if some of the documents are visited.

For example, in Figure 4.5, site 3 is marked as '1' since all the documents in that site are

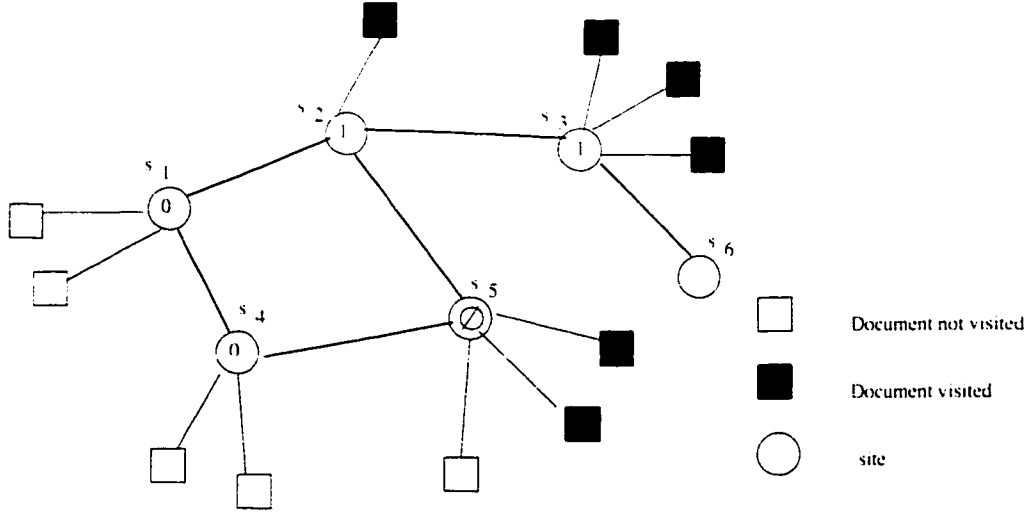


Figure 4.5: Site-Document(Marked) graph

visited. Site 5 is marked as '⊙' since only some of the documents in that site are visited and site 4 is marked as '0' since no documents of the site are visited.

#### 4.2.2 Case study

Consider a situation where 3 agents  $A_1$ ,  $A_2$  and  $A_3$  are present in a work group. A Site-Document graph  $G_1$  (Figure 4.6) is generated corresponding to a query  $Q_1$  by the user  $U_1$ . The user interface agent  $A_1$  helps the user in developing a navigation plan. The agents  $A_2$  and  $A_3$  belong to the same work group and have executed the query  $Q_1$ , and hence they have the knowledge of Site-Document graph  $G_1$ . The agent  $A_1$  gets the help of agents  $A_2$  and  $A_3$ , to develop a navigation plan for  $G_1$ .

The agent  $A_1$  gets the help of agents  $A_2$  to arrive at a navigation plan, visit nodes in the order 2, 3, 4, 5 and 6, for the query 'Get all Architecture images by the author Ramsay Traquair'.

*Note:* In the site-document graph  $G_1$  in figure 4.6 nodes 7, 8 and 9 are eliminated in the plan suggested by  $A_2$ .

A sample interaction which can take place between  $A_1$  and  $A_2$  to obtain the plan is show

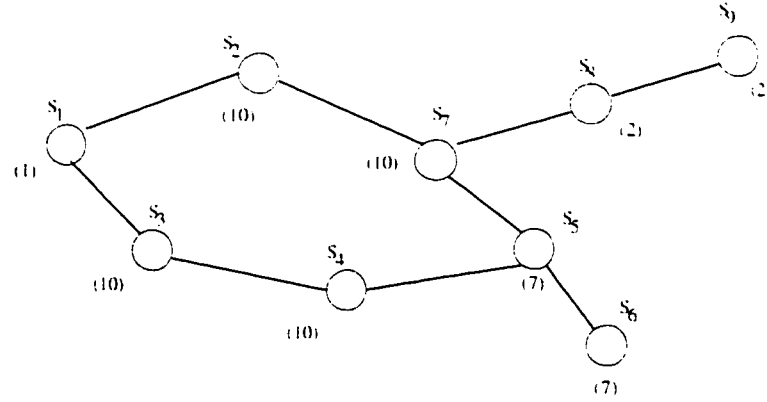


Figure 4.6: Site-Document graph  $G_1$  for query  $Q_1$ . Circles represents the different sites and the arch represents the link between them, and the number within the bracket represents number of documents in the site which match the query.

below.

1.  $A_1$ : Can you give me a proposal for navigating the graph  $G_1$ .
2.  $A_2$ : The sites 2, 3, 4 and 7 have ten documents and, site 5 and 6 have seven documents each related to the 'Author = Ramsay Traquair'. The sites 8 and 9 have only two documents each related to the same author. So you can visit the sites 2, 3, 4, 5 and 6.
3.  $A_1$ : What if I visit 2, 3, 4, 5 and 7? What will I loose?
4.  $A_2$ : Though site 7 has ten documents related to the 'Author = Ramsay Traquair', only two of them are in the field of architecture. Hence, visiting site 6 instead of 7 will be useful.
5.  $A_1$ : O.K.

In the utterance (2) above,  $A_2$  proposes a plan to ' $A_1$ ', which include visiting sites 2, 3, 4, 5 and 7 based on its private belief.  $A_1$  gives a counterproposal (3), visiting site 7 instead of site 6.  $A_2$  points out the disadvantage of  $A_1$ 's proposal and provides evidence for the original proposal (4).  $A_1$  re-evaluates  $A_2$ 's proposal, which consists of beliefs conveyed by utterance 2 and 4, and accepts  $A_2$ 's proposal.

The above assumption holds good for  $A_3$ .  $A_1$  has a similar conversation with  $A_3$  and gets a plan.  $A_1$  now will compare  $A_2$ 's proposal with that of  $A_3$ 's proposal, to find if there is a conflict. If there is a conflict, it can negotiate with  $A_2$  and  $A_3$  to resolve the conflict or find the best plan.

#### **4.2.3 Algorithm for cooperative navigation (Case - 1: two agents situation.)**

The Algorithm for cooperative navigation is constructed based on the following assumptions:

- The agents are honest i.e.; they are not lying and are consistent in their error.
- The agents are experts in their respective domains, because their suggestion is simply accepted.
- The agents are co-operative i.e.; Two agents are said to be cooperating with each other in the sense that they respond to a request as per Table 4.1.

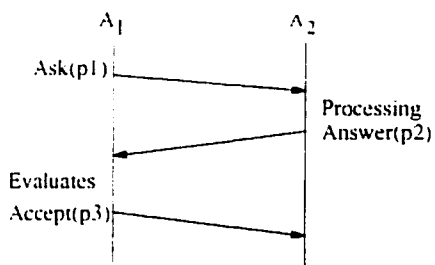
**Algorithm (Figure 4.7): protocol for exchange of messages.**

1. An agent  $A_1$  (initiator) uses 'Ask (p1)' speech act to get a plan from agent  $A_2$  for navigating a marked graph. P1 consists of a site-document graph and information about the present site ( $S_c$  and  $D_c$ ) of agent  $A_1$ .
2. Agent  $A_2$  responds using 'Answer (p2)' speech act, which consists of a vector  $V_{21}$  for  $A_1$  to navigate in the site-document graph.

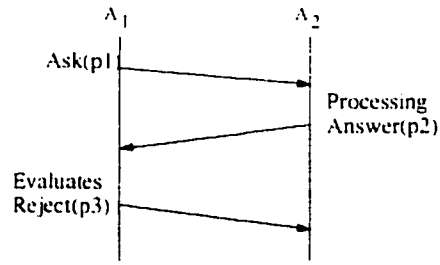
Note: Agents arrive at vector  $V_{ij}$  by using its knowledge of the graph, and using the navigation functions  $S()$ ,  $D()$  and choice function  $C()$ .

3.  $A_1$  evaluates  $A_2$ 's proposal and selects one of the following:

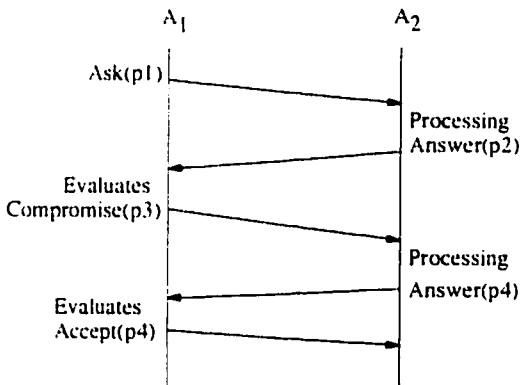
- (a) If the agent  $A_1$  is satisfied with the response, it accepts the proposal and sends 'Accept (p2)' to agent  $A_2$ .



(a)



(b)



(c)

- p1: Consists of site-document graph and information about present site
- p2: Consists of vector  $V_{sp}$ .
- p3: Counter proposal
- p4: Vector  $V_{sp}$  and evidence in support of the vector

Figure 4.7: Protocol diagram for two agent situation

- (b) Else, if agent  $A_1$  has some clarification, it uses 'Compromise (p3)' speech act to clarify about agent  $A_2$ 's proposal, wherein p3 consists of a counter proposal. Counter proposal is limited to once for the sake of simplicity.
  - (b1) Agent  $A_2$  responds with 'Answer (p4)', which can include evidence in support of the proposal. Go to step 3.
- (c) Else, Agent  $A_1$  rejects the proposal using 'Reject (p)' speech act. This can be due to agent  $A_1$  not accepting the agent  $A_2$ 's belief or the evidence provided by the agent.

#### 4.2.4 Detecting conflict and resolving conflicts by negotiation

There are number of reasons why conflict may arise. For instance, agents may have "knowledge conflict" due to problems of incompleteness, uncertainty or non reliability in their own knowledge databases. Negotiation is suggested by many researchers as an important method to resolve conflict, which can be defined as *a process of communication among agents in which conflicting goals are reformed; conflicting issues are identified and narrowed; alternative solutions are proposed, attacked, and defended; and an agreement is reached and confirmed* [PP87]. Hence the objective behind negotiation is to reach a consensus.

##### **Example 1 : Example of a conflict**

*Consider a three agent situation (Figure 4.9). Agent  $A_1$  is trying to find a navigation plan by consulting  $A_2$  and  $A_3$ . If it gets conflicting proposals, it will first try to detect the source of conflict by the following method:*

*Agent  $A_1$ , using the algorithm described in the previous section gets a proposal from  $A_2$  and  $A_3$ , which constitutes vector  $V_{21}$  and vector  $V_{31}$  respectively (Figure 4.8). In this  $V_{ij}$  is the response from agent  $A_i$  to agent  $A_j$ .  $A_1$  now compares  $V_{21}$  with  $V_{31}$ .*

*To detect the conflict in the proposal, set (S) equivalent of the Vector (V) is compared.*

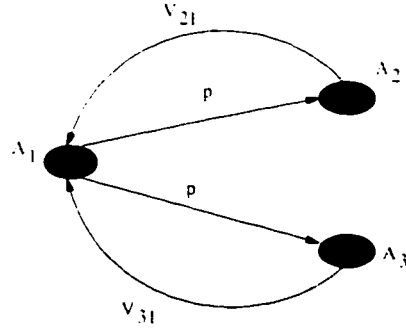


Figure 4.8: Communication between agents

*If  $S_{21} = S_{31}$ , there is no conflict in the proposal.*

*Else, if  $S_{21} \neq S_{31}$ , there is a conflict in the proposal.*

To resolve a conflict, first the focus of the conflict is determined. Once it finds the focus of the conflict, the agents can negotiate or use confidence function ' $\otimes$ ' to resolve the conflict. The confidence function is based on an agents knowledge of the other agent and/or previous experience with that agent, which can be a weighted function (Example: Figure 4.10).

If  $(\otimes_{A_2} > \otimes_{A_3})$ , then accept  $A_2$ 's proposal.

Else, if  $(\otimes_{A_2} < \otimes_{A_3})$ , then accept  $A_3$ 's proposal.

Else, if  $(\otimes_{A_2} = \otimes_{A_3})$ , then use 'Compromise (p3)' speech act once (for simplicity) with  $A_2$  and/or  $A_3$  to resolve the conflict or to find the best plan.



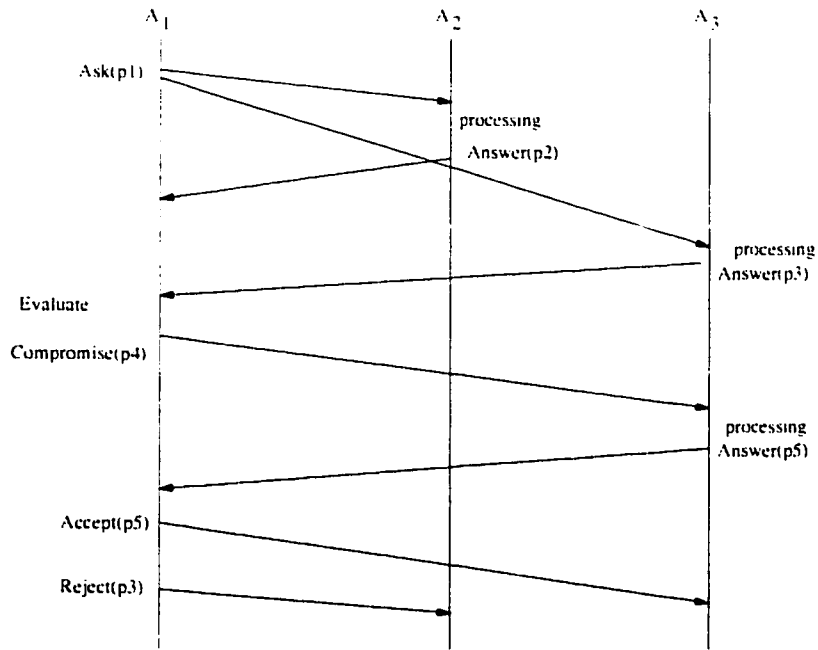


Figure 4.9: Protocol for three agents situation. In this  $A_1$  gets a proposal from  $A_2$  and  $A_3$ . It then compares the proposal. Since it is not equal, it uses compromise act once with  $A_3$  to find the best proposal.

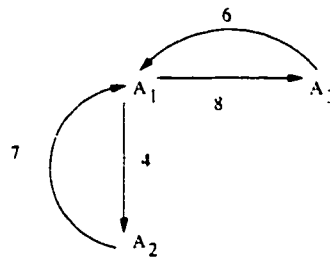


Figure 4.10: Directed graph showing the confidence function. The weightage is between 1-10. The number in the graph represents the weight of confidence.

### **4.3 Summary**

In this chapter, we have investigated cooperative navigation in digital library as a method to support the sharing of information processing knowledge within a team of users and also address the challenges and opportunities that the transition to digital libraries affords.

## Chapter 5

# Implementation Overview

In this chapter, we provided an overview of our implementation of the User interface for a Digital library that includes both text and images. We focus on the programming aspects of user registry, query formulation and search result retrieval, and provide a walk through of GUI using Figures and screen-captures. The implementation environment was the following:

- Solaris 2.5 on Sun Ultra-1 creator.
- JDK 1.1.7 (Java Development Kit).
- Swing1.1.beta3 (Java Foundation classes for the GUI).
- Emacs with JDE Java wrapper.

### 5.1 Implementation of the User interface

Following is a list of files of our implementation and a brief description of their functions.

- UIDigitalLibrary: Main Window of the graphical user interface.
  - menuPanel: This function creates the main menu.
  - titlePanel: This class creates the title page.
  - attributePanel: GUI for attribute search.
  - similarityPanel: GUI for similarity search.

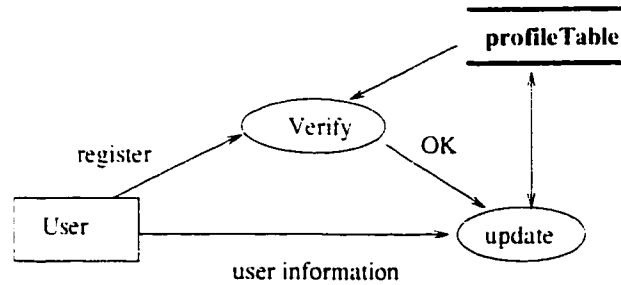


Figure 5.1: Functional model for registration

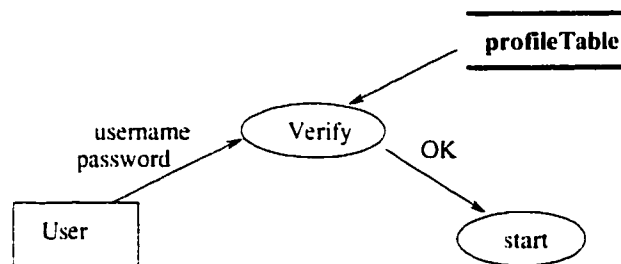


Figure 5.2: Functional model for verifying the password

- verifier (explained in section 5.1.1).
- query (explained in section 5.1.2 ).
- resultPanel (explained in section 5.1.3 ).

### 5.1.1 Registering the user

Every first-time user has to register with the system so that the system gets information for the user profile and assigns a valid password and user name. The functional model shown in Figure 5.1 and 5.2 depicts which of the values depend on which other values and the functions that relate them. In the functional model, a square box represents external object, such as User in Figure 5.1; two horizontal lines, such as profileTable in Figure 5.2 represent attribute; oval represents the functions, such as ShowRandomURLs in Figure 5.4.

The verifier class contains the following attributes and methods:

- Attributes:
  1. profileTable - profileTable is a hash-table and contains (password , link(to the profile table)) value pair.

- 2. PasswdTab - It is a hash-table containing username and password.
- 3. password - Contains the value of the user password.
- 4. userName - Contains the user name.
- methods:
  1. verify - This method gets the information from the user and calls update function to update the profile. If the user has already registered it checks if the user name and the password are valid.
  2. loadData - Loads the users personal information.
  3. update - Updates the user profile.

### **5.1.2 Query formulation**

A query formulated by the user is submitted to the search engine by passing Query objects. The query object contains the criteria that is used to retrieve items from the database. In addition, it also contains certain site properties.

The query class contains the following attributes and methods:

- Attributes:
  1. Query - The query is either a string or a parsed tree.
  2. siteDescription - The information related to various site details
- Methods:
  1. GetSummary - Get the QuerySummary. A QuerySummary is all the information in the Query object packaged into a class which can be transmitted to the search agent.
  2. GetQueryDescription - Get just the query string or parsed query tree.
  3. SetQueryDescription - Set the QueryDescription to a new value.
  4. GetSiteNames - Get the various site related attribute.
  5. SetSiteNames - Set the siteDescription attribute to a new value.

#### **5.1.2.1 Parsed Query Description Format**

The RPNQuery class defines QueryDescription data structure which contains the parsed form of the query. The implementation of this data structure assumed Z39.50 standard in Reverse Polish Notation (RPN) format. The tree structure is generic and flexible. A

QueryDescription can be in either of two formats as: a tree or as a string. Therefore, the root of the tree is a discriminated union which specifies the format. The "Query Type" value specifies which structure is used.

RPNQuery0 class contains the following:

- **Attribute:**

1. **attributeSet:** The attributeSet describes what attribute set is used in this tree. The operators in the tree may have attributes associated with them. The set of values from which these attributes are drawn is defined by the attributeSet.
2. **RPNOperator:** RPNOperator is a set of operators, wherein, three boolean operators (AND, OR, and NOT) are supported.
3. **RPNOperend:** RPNOperend is a hashtable which contains Type and Value pair. This is used to indicate the value assigned to an attribute.

- **Methods:**

1. **NodeStructure** - NodeStructure determines the nodes of the tree. A node can be either a RPNOperand or an RPNOperator with an associated sequence of nodes which it operates on.  
Leaf node: One possibility for an NodeStructure is that it may be a leaf node, i.e., an operand, which may be a sequence of attributes/Value pair. Internal node: The other possibility for an NodeStructure is that it may be an internal node, i.e., it is an operator, which is a set of boolean operator.
2. **isValue** - determines the operator.

### 5.1.3 Retrieving the search results

The functional model of ResultPanel class is shown in Figure 5.3. This class provides the sequence of operations that are performed by the search engine when retrieving the search results.

The methods used for retrieving the search results are:

1. **SearchEngine** - This method processes the Finalquery. (implemented by University of Montreal, as a part of the digital library project)
2. **GetTotalNumberOfItems** - This method returns the number of items that was returned after the search has been performed.
3. **RetrieveItems** - This method retrieves the search results (list of URLs).
4. **addtoList** - It creates a list of items returned.

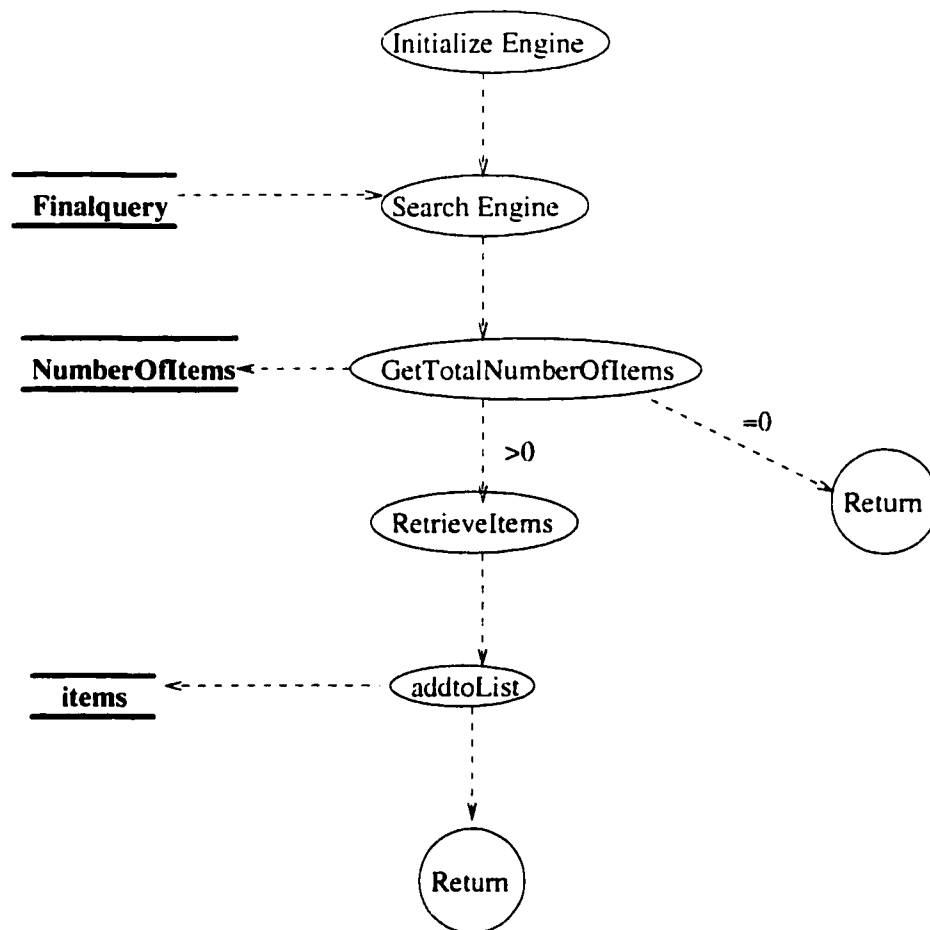


Figure 5.3: Functional model for retrieving the search results

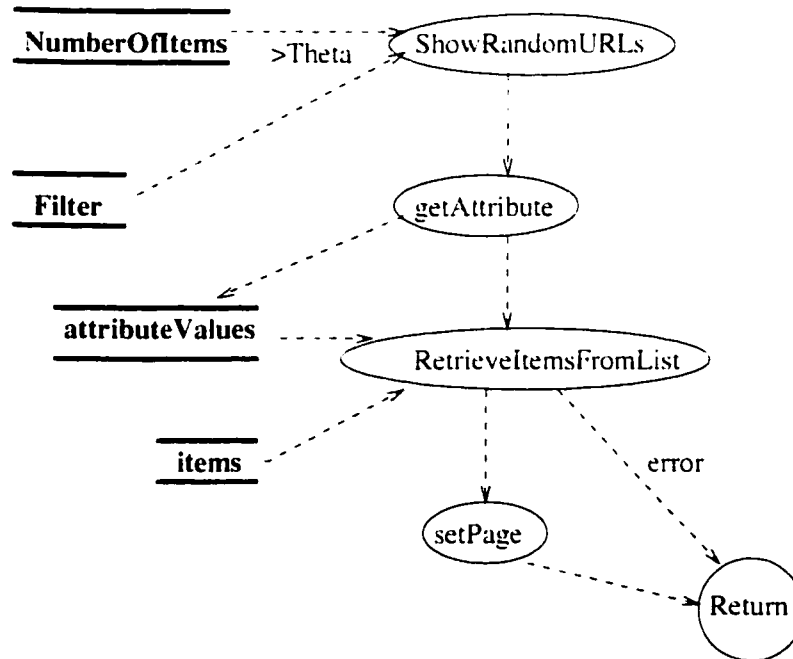


Figure 5.4: Functional model for filtering

#### 5.1.4 Filtering and displaying the search results

The functional model of the filtering and presentation is shown in the Figures 5.4, 5.5 and 5.6. The methods perform a list of operation when filtering and displaying the search results.

The methods used for filtering are:

1. ShowRandomURL - This method shows the user a list of random URLs from which the user can select attribute values for filtering. (Implemented by the child class of ResultPanel).
2. getAttribute - This method gets the attribute values from the user (Implemented by the child class of ResultPanel).
3. RetrieveItemFromList - This method returns a list of URLs after filtering.
4. SetPage - This creates a page to view the URLs sequentially.



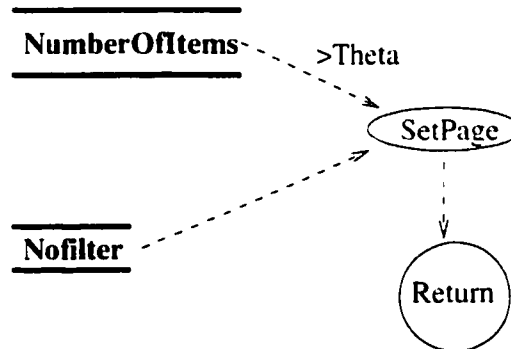


Figure 5.5: Functional model for displaying the URLs sequentially without filtering

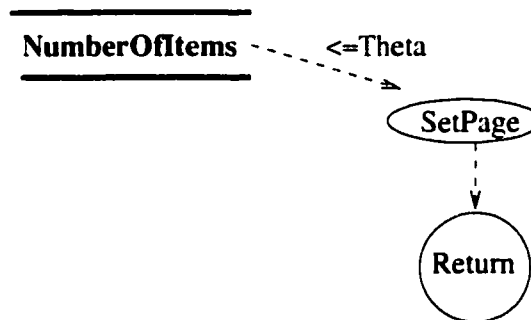


Figure 5.6: Functional model for displaying the URLs sequentially

## 5.2 Description of the GUI

The main goal of our user interface is “ease of use” for a digital library search.

A new user starts by registering first. From the browser window, the user should click the registry button. A window will open where the user can give preferred user name and password, detail about the user’s preferences and personal interest.

The user can now start using the digital library with the User Name and password. Password verification results will be displayed on the message board. If the verification fails, user has to attempt again giving the correct user-name and password. The user will be able to proceed once the verification succeeds.

The opening window of the user interface for digital library is shown in Figure 5.7. Two categories of search interfaces are supported in our system, namely attribute based search and similarity based search. The user can select a required search using the appropriate tab button in this window.

Attribute based search shown in Figure 5.8, lets the user enter words and phrases, select the field such as title, location, photograph and building types, to be searched, and choose Boolean operators from pull-down menus. In this interface, the user can search for single words or phrases, or the “AND/OR” of two fields. Search can be further refined by specifying the domain of search (from the “Domain” list), type of database to be searched (from the “Type of Database” list) and database name (from “Database Name” list).

Similarity based search shown in Figure 5.9, lets the user select an image from a list of images given in the interface. In this interface, lists of four images are displayed for each building type. The user can select the building type and also the type of indexing used.

The user interface will process the user input and generate as its output, a Z39.50 type1 query. Type 1 query is based on Reverse Polish Notation (RPN), which defines an expression comprised of operators and operands. The query will be displayed in the query text box

(5.10), when the user selects “Query” option from the pull down menu labelled “Search”. The user can then edit his query if he wants to. The final output from the user interface will be sent as an input to the search engines.

### **5.2.1 Response-set presentation**

The result set interface for Digital library is shown in Figure 5.13 . This interface has a panel for viewing list of URLs and a panel for displaying the HTML document and then to follow their links.

The List panel will display the query result as a list of URLs. The user can click on the URL to go to a particular document site. The selected document is displayed using the HTML panel. Further, the user can navigate between documents by using “Back” button and “Search results” button.

The presentation agent uses filter to remove unwanted information from the response set. Filtering is done when the response set is too large to browse sequentially. The filtering is based on the user profile and the content of the response set.

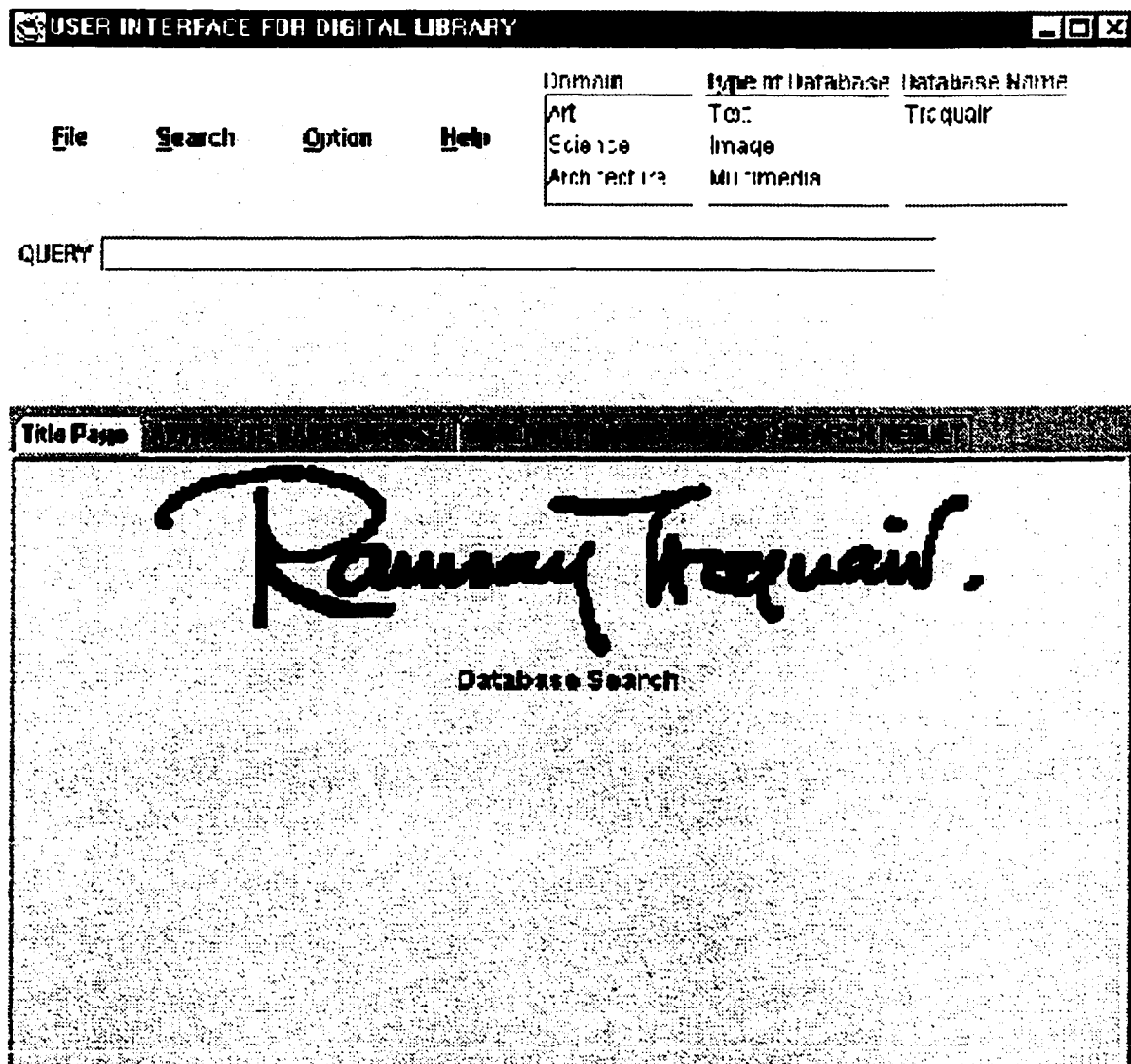



Figure 5.7: Opening window of the Digital Library


**USER INTERFACE FOR DIGITAL LIBRARY**

File

Search

Option

Help

Domain	Type of Database	Database Name
Art	Text	Tricquair
Science	Image	
Architecture	Multimedia	

QUERY

ATTRIBUTE BASED SEARCH

Search Photograph by:

Contains the Keywords

AND

Title

Contains the Keywords

AND

Photographer

Harvey, Edgar

AND

Building types

Commercial

Apartment

Major Type

Minor Type

Figure 5.8: Interface for attribute based search



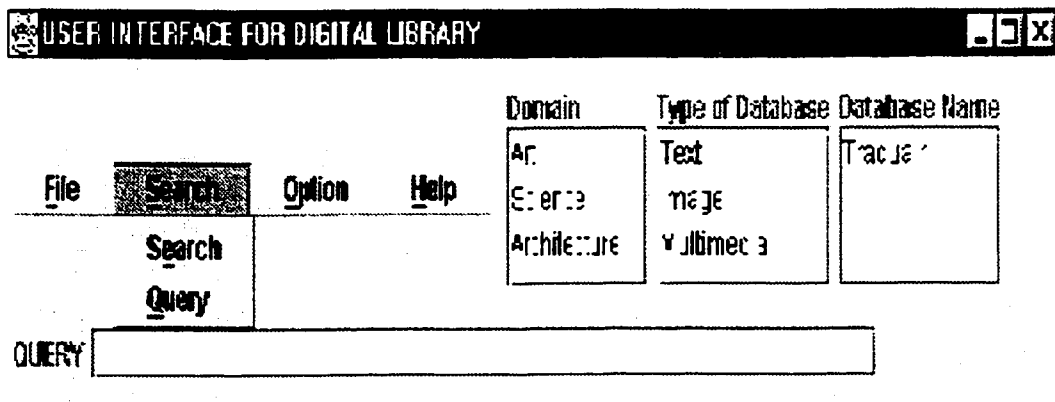


Figure 5.10: Search menu

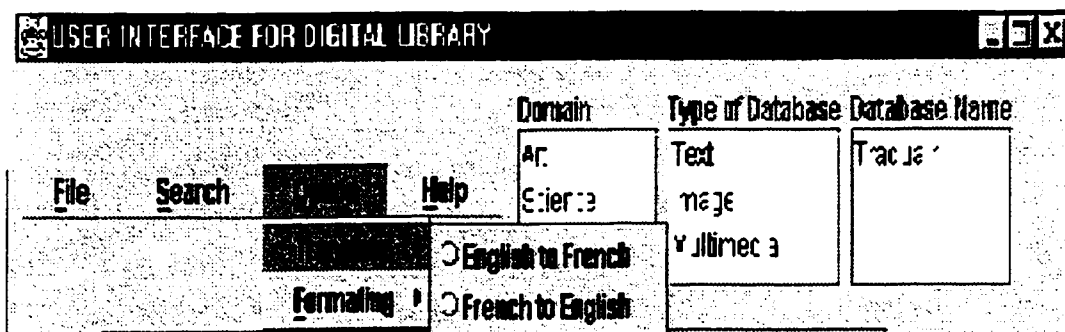


Figure 5.11: Option menu

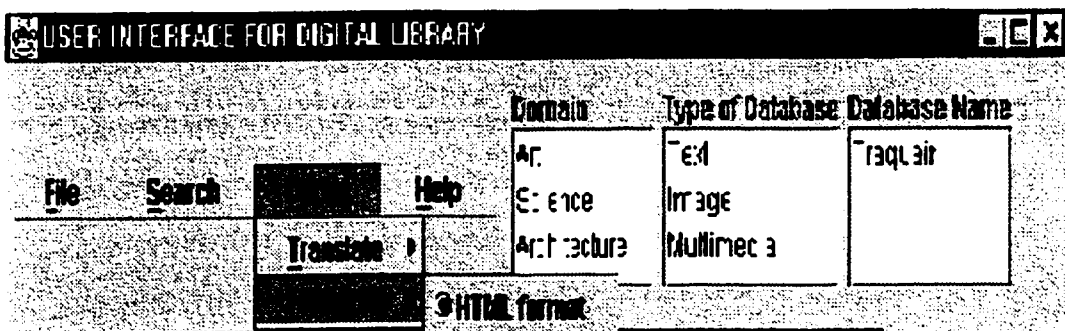


Figure 5.12: Option menu

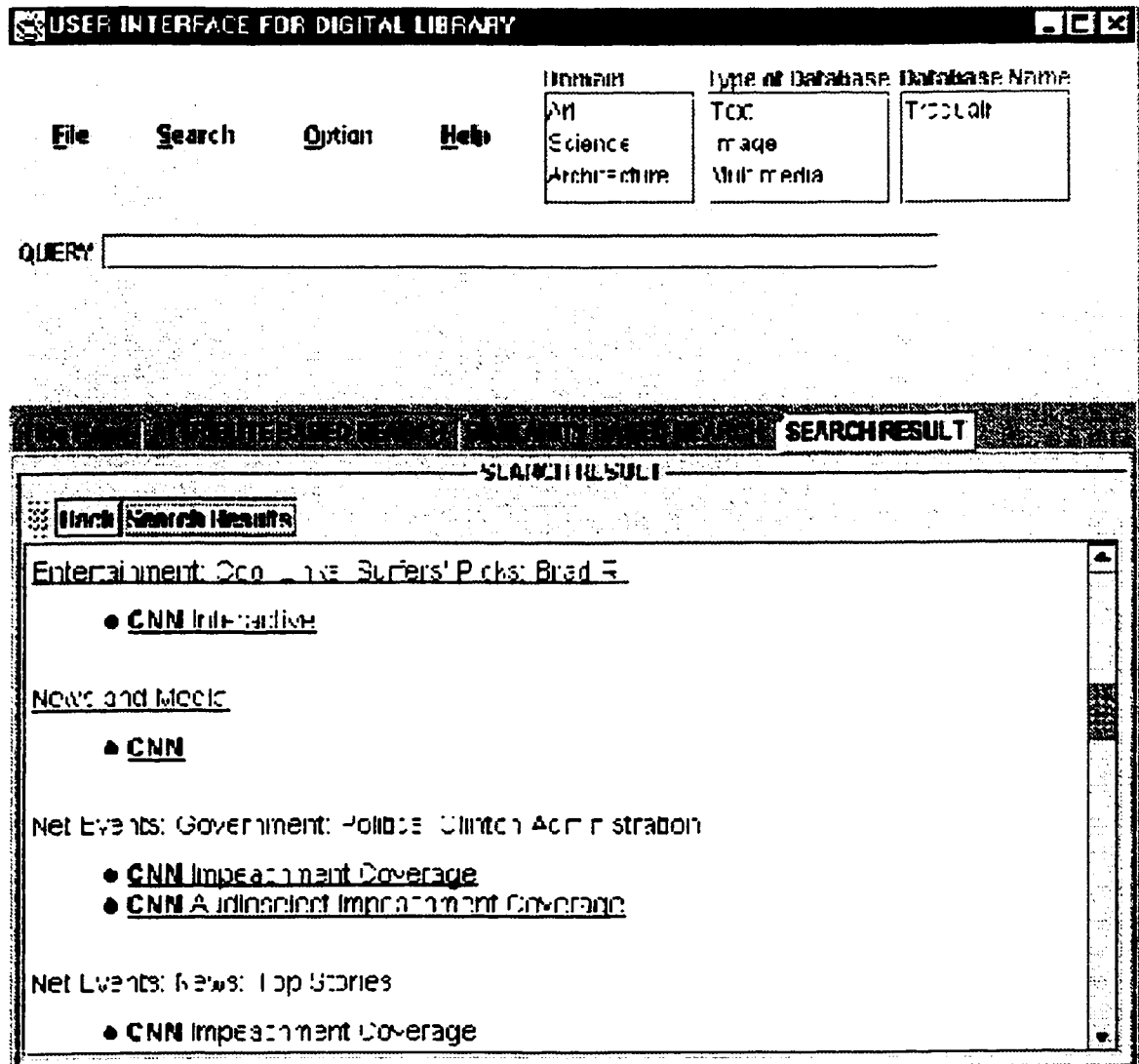


Figure 5.13: Interface for displaying the search result



## Chapter 6

# Conclusion

An easy to use and versatile user interface prototype is developed as a part of this thesis for a joint project undertaken with three other Montreal based universities. This prototype provides a uniform interface for texts and images. It makes use of a simple conceptual model that is well suited for digital libraries with a set of affordances for interacting with the interface. The architecture allows users to access the interface and display the results under a web browser. It also has a filter, which filters the search results using user profile when there are 1000's of items retrieved as a response.

This chapter revisits the goals that were specified in Chapter1, and describes how the thesis meets each goal. It finally concludes with a discussion of some of the issues raised by this work, and suggests future directions for addressing those issues.

### 6.1 Goals revisited

Based on the needs of the users of digital libraries, as gleaned from the studies of traditional library users, two main goals were set out in Chapter1 in the design of user interfaces to digital libraries. Those goals were to:

- Support user's tasks and
- Sharing and reusing information gathered.

This section reviews the work in terms of those goals.

### **6.1.1 Support user's tasks**

The first goal for any user interface to a digital library is to support users in the tasks they need to accomplish. This goal stems out of the context of traditional information retrieval systems, which assume that searching by itself is the activity of interest, and tends to provide little or no support beyond the presentation of results.

A working prototype developed as part of this thesis supports tasks by providing users with various types of search interface, which contain resources appropriate to task at hand and visually indicate the state of the current task. Bates [Bat89] points out "*information needs change throughout a series of searches and the searches need an accumulation of results rather than a single target result*". In agreement with the above statement, this prototype provides a mechanism to store and retrieve intermediate search results. Further it also provides a filter to bring the result set closer to match the user's information needs. This filter is based on user profile and feedback from the user on the relevance of terms appearing in the result set. Lastly, this prototype provides a starting point for support of user tasks, especially those tasks that require the use of different document types, such as text and image.

### **6.1.2 Sharing and reusing information gathered**

Information retrieval systems have been largely designed to give the impression of being single user systems i.e., the existence and activities of other users have been hidden from each other. In this thesis, we argue that introducing support for reusing and sharing information

gathered among multiple users in information retrieval systems would help them to use the systems more effectively.

The thesis, addresses the above problem through the use of user agents. In this case, there are several challenges that require careful consideration and analysis. In this thesis, we describe an agent framework for sharing and reuse of search results. An user agent communicates with other user agents using five speech act primitives. A protocol is proposed for cooperation and “intelligent” inter-operation between agents. This thesis examines how conflicts can arise in the cooperative endeavor and how they can be resolved using negotiation. Two main functions of the system have been identified, including helping user find the relevant information that meets his specific needs.

## **6.2 Discussion**

A working prototype is implemented and tested. It makes use of an image database (Traquair photographic archive) developed by McGill University as a part of the inter-university project. The design of this prototype has addressed the stated goals for a digital library interface.

Indeed, the next step will likely be to perform user studies of the prototype. If the prototype were to be refined to be a polished product, that refinement should include performance improvements in the Java client, and regular updates to keep it synchronised with the latest versions of Java and web browsers.

Main issues related to sharing and reuse of information gathered are privacy and ownership, which this thesis has not addressed. Librarians have a long tradition of concern for the privacy of their users. Collaboration usually involves some reduction of this privacy. When the collaboration is with known individuals, the users can freely choose the degree to which they make personal activity and information available. However an electronic system offers

the opportunity (should people want it) of collaboration with strangers. This may be achieved in part by the system identifying similar research interests and offering to introduce the participants to each other, leading to a conventional form of collaboration. The second problem of ownership arises in which the question of rights regarding search history and its use in future has to be resolved.

It would seem, that to be acceptable to users, a system supporting collaboration should make clear to users the benefits that accrue from their loss of privacy along with precise details about the way in which information about their search activities are, and may be, used. Future work on this proposal should include these issues.

# REFERENCES

- [All87] J. Allen. *Natural Language Understanding*. Benjamin/Cumming Publishing Company, Menlo park, CA, 1987.
- [Aus62] J.L. Austin. *How to do things with words*. Oxford University Press: Oxford, England, 1962.
- [Bat89] M. Bates. The design of browsing and berry picking techniques for online search interfaces. Technical report, Online Review, 1989.
- [BP88] M.E. Bratman and M.E. Pollack. Plans and resource-bounded practical reasoning. In *Computational Intelligence*, pages 349-355, 1988.
- [BW94] P. W. Birmingham and K. Willis. The university of michigan digital library: This is not your father's library. In *Proc. Digital Libraries*, 1994.
- [CL90] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. In *Artificial Intelligence*, pages 213-261, 1990.
- [DM96] E. A. Daniel and P. W. Michael. Towards inquiry-based education through interactive software agents. In *IEEE Computer*, pages 69-79, 1996.
- [Eic94] D. T Eichmann. The rbse spider – balancing effective search against web load. In *proceedings of the First International Conference on the World Wide Web*, pages 369-378, Geneva, Switzerland, 1994.

- [FH85] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. In *proceedings of the ninth International Joint Conference on Artificial Intelligence (IJCAL-85)*, pages 480-490, Los Angeles, CA, 1985.
- [GK94] M.R. Genesereth and S.P. Ketchpel. Software agents. In *Comm. of the ACM*, pages 48-53, 1994.
- [GL87] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *proceedings of the Sixth National Conference on Architecture Intelligence (AAAAI-87)*, pages 667-682, Seattle, WA, 1987.
- [GRSR97] P.C. Gokul R. Shinghal and T. Radhakrishnan. Designing cooperating agents for office automation. In *Fourteenth National Conference on Artificial Intelligence (AAAI 97): Workshop on Using AI in Electronic Commerce, Virtual Organizations and Enterprise Knowledge Management to Reengineer the Corporation*, pages 33-39, Providence, Rhode Island, 1997.
- [GT94] D. Goldberg and D. Terry. Using collaborative filtering to weave an information tapestry. In *Communications of the ACM*, pages 61-70, 1994.
- [HF95] W.C. Hill and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *proceedings of the Conference on Human Factors in Computing Systems (CHI'95)*, pages 194-201, Denver, CO, ACM, 1995.
- [Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press: Ithaca, NY, 1962.
- [HS95] C. Huser and N. Streitz. Knowledge-based editing and visualization for hypermedia encyclopedias. In *Communications of the ACM*, pages 49-51, 1995.
- [Hya96] S. N. Hyacinth. Software agents: An overview. In *The Knowledge Engineering Review*, 1996.

- [Inc87] Apple Computer Inc. *Human Interface Guidelines: The Apple Desktop Interface*. AddisonWesley Publication Co., Reading, MA, 1987.
- [Joh89] K. Johnson, J. & Mackey. The xerox star: A retrospective. In *IEEE Computer*, pages 11-30, 1989.
- [KM93] R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. In *proceedings of the ACM SIGCHI International Workshop on Intelligent User Interfaces*, pages 81-93, Florida, 1993.
- [Kri63] S. Kripke. Semantical analysis of modal logic. In *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, pages 67-97, 1963.
- [Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *proceedings of the Fourth national conference on artificial intelligence(AAI84)*, pages 198-202, Austin, TX, 1984.
- [Lid96] D. Liddle. *Design of the Conceptual Model*. In Winograd T (ed), *Bringing Design to software*. (Reading, MA: Addison Wesley, 1996.
- [Lie95] H. Lieberman. Letizia: An agent that assists web browsing. In *proceedings of IJCAI*, 1995.
- [Mae] P. Maes. <http://pattie.www.media.mit.edu/people/pattie/>.
- [Moe95] William E. Moen. Z39.50: Information retrieval(z39.50): application service definition and protocol specification. Technical report, National Information Standards Organization, 1995.
- [PP87] L.L Putman and M.S. Poole. Conflict and negotiation. In *the handbook of organization communication: An Interdisciplinary perspective*, F.M. Jablin et al., Eds. Sage, Newbury Park, Calif.,, pages 549-599, 1987.

- [PR90] M. E. Pollack and M. Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *proceedings of the Eighth National Conference on Artificial Intelligence (AAAI90)*, pages 183-189, 1990.
- [RG91] A. S. Rao and M.P. Georgeff. Modeling rational agents within a bdi-architecture. In *proceedings of Knowledge Representation and Reasoning*, page, pages 473-484, 1991.
- [Rhe96] E. RheinfrankJ. Design languages. In *In Winograd T, Bringing Design to software, (Reading, Mass. : addison Wesley*, pages 63-80, 1996.
- [SB96] M. Salampasis and C. Bloor. Co-operative information retrieval in digital libraries. Technical report, presented at the 18th annual colloquium of the BCS IR SG, Manchester, U.K, 1996.
- [Sea69] J. Searle. *Speech Acts*, Cambridge. MA, Cambridge University Press, 1969.
- [urla] <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/pleiades.html>.
- [urlb] <http://www.genmagic.com>.
- [VB90] S. Vere and T. Bickmore. A basic agent. In *Computational Intelligence*, pages 41-60, 1990.
- [WJ98] M. Wooldridge and N. Jennings. The pitfalls of agentoriented development. In *proceedings of the Second Conference on Autonomous Agents(Agents'98)*, 1998.