

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[®]
800-521-0600

User Interface Agents in Electronic Commerce Applications

Ji Lu

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

December 1999

© Ji Lu, 1999



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.

0-612-48303-7

Canada

ABSTRACT

User Interface Agents in Electronic Commerce Applications

Ji Lu

A User Interface Agent (UIA) is a software entity that assists its users in effectively using the user interface software. In electronic commerce applications, the user interface software plays an important role. Users of different kinds and backgrounds will use this software with different expectations. Traditional studies of user interface technology have shown the existence of a “gap” between what the user interface software can actually do and the users’ expectations about it. Agent technology, in the form of personalized user interface agents, can help narrow this gap.

In this thesis, the supports that software agents can provide in various stages of consumers’ shopping process are introduced. Roles of the user interface agent and contents of the user model are analyzed. Also, this thesis explores the use of a user interface agent to provide personalized service to the user by knowing the user’s preferences. The UIA maintains a knowledge base about the user, based on a user model. Such user models can be constructed by taking application dependent and application independent attributes into account. The user model is used to deal with incompleteness and ambiguity that might arise in the user’s search query, to help the UIA decide when and what kind of proactive information to present to the user, or to enable the UIA to filter the search response set. The UIA assists the user by “reasoning” effectively based on a set of rules that captures the user’s needs, preferences and the way he is interacting with the software system. A prototype for UIA has been implemented on the Java platform for electronic shopping. A set of test cases demonstrates that employment of the user model for the UIA improves the effectiveness of the user interface in electronic commerce applications.

Acknowledgements

Sincerely, I would like to express my deepest respect and gratitude to my thesis supervisor Dr. T. Radhakrishnan for his guidance, invaluable suggestions, encouragement and support throughout the course of this research work. I am very thankful for the opportunity, I had, to work with Dr. T. Radhakrishnan, and for everything he taught me during my master's studies.

I wish to thank my colleagues at the Multimedia Laboratory of Concordia, especially Patrick, Venkat and Rony for the fruitful discussions, suggestions and comments on the project and software tools.

I would like to give a special thanks to my husband, my lovely son and my mother. It was their love, patience and continued support that made this thesis possible.

Financial support of this project provided by the Canadian Institute for Telecommunications Research through a research grant to Dr. T. Radhakrishnan is greatly acknowledged.

Table of Contents

LIST OF FIGURES	VII
LIST OF TABLES	IX
1. INTRODUCTION.....	1
1.1 OVERVIEW OF ELECTRONIC COMMERCE	1
1.2 AGENTS AND USER INTERFACE AGENTS.....	3
1.3 OVERVIEW OF THE CITR PROJECT	5
1.4 OBJECTIVE AND SCOPE OF THE THESIS	6
1.5 ORGANIZATION OF THE THESIS	7
2. AGENT TECHNOLOGIES FOR E-COMMERCE	9
2.1 USERS' SHOPPING BEHAVIORS IN E-COMMERCE.....	9
2.2 AGENTS AS MEDIATORS IN E-COMMERCE	11
2.2.1 <i>Product Brokering</i>	11
2.2.2 <i>Merchant Brokering</i>	13
2.2.3 <i>Negotiation</i>	14
2.3 FUTURE DIRECTIONS	15
3. USER INTERFACE AGENTS AND THEIR APPLICATIONS	17
3.1 ARCHITECTURE AND ROLES OF UIA.....	17
3.2 USER MODELING	19
3.2.1 <i>Contents of User Model</i>	20
3.2.2 <i>Classification of User Models</i>	20
3.2.3 <i>Building and Updating User Model</i>	22
3.3 APPLICATION OF INTELLIGENT INTERFACE AGENTS.....	23
3.3.1 <i>Case Study 1: A UIA for Electronic Mail Handling</i>	26
3.3.2 <i>Case Study 2: IR-NLI II — A UIA for Intelligent Information Retrieval</i>	28
3.3.3 <i>Case Study 3: User Modeling in A Commercial Software System TIMS</i>	32
3.3.4 <i>Summary</i>	36
3.3.5 <i>Comparisons</i>	38

4. A UIA PROTOTYPE FOR E-COMMERCE	41
4.1 CITR SUBPROJECT – USER INTERFACE AND INTELLIGENT AGENT	41
4.1.1 <i>User Interface Agent for E-Commerce</i>	42
4.1.2 <i>Software Sales Agent for Price-Negotiations</i>	43
4.2 USERS' SHOPPING BEHAVIORS	44
4.3 PROTOTYPING THE UIA	46
4.3.1 <i>Objective of the UIA Prototype</i>	46
4.3.2 <i>Functionality of the UIA Prototype</i>	46
4.4 USER MODEL FOR PROTOTYPING THE UIA.....	49
4.5 PRIVACY OF USERS' PERSONAL INFORMATION.....	53
4.5.1 <i>Protection of the user's personal information</i>	53
4.5.2 <i>Technical Issues</i>	54
5. IMPLEMENTATION AND TESTING.....	56
5.1 IMPLEMENTATION ENVIRONMENT	56
5.2 SOFTWARE ARCHITECTURE	56
5.3 SUBSYSTEM DESCRIPTION	58
5.3.1 <i>User Profile Management</i>	58
5.3.2 <i>Database Access and SQL Generation Utilities</i>	62
5.3.3 <i>Result Presentation</i>	66
5.3.4 <i>Shopping Cart Metaphor</i>	69
5.3.5 <i>Jess Engine and Rules</i>	71
5.4 TESTING OF QUERYING AND UIA ASSISTANCE	73
5.5 TESTING OF SHOPPING CART METAPHOR.....	75
6. SUMMARY AND CONCLUSIONS.....	77
6.1 CONCLUSIONS.....	77
6.2 FUTURE WORK.....	79
7. REFERENCES.....	81

List of Figures

Figure 1: Four knowledge acquisition sources for learning interface agents.....	18
Figure 2: Dimension of user models [Dieterich 93].....	21
Figure 3: Knowledge acquisition (adapted from [Dieterich 93])	22
Figure 4: Reference architecture for information filtering agents.....	24
Figure 5: Reference architecture for information retrieval agents	25
Figure 6: Reference architecture for intelligent assistants.....	25
Figure 7: Architecture of IR-NLI II (adapted from [Brajnik 87])	29
Figure 8: The User Modeling Component architecture of TIMS [Strachan 97]	34
Figure 9: The basic algorithm for updating user model [Strachan 97].....	35
Figure 10: Functionality of the UIA.....	47
Figure 11: User interface pointing to other components.....	48
Figure 12: User information to build a user model	50
Figure 13: Rules for user's choices or preferences in the user model	51
Figure 14: Rules for facts in the user model	51
Figure 15: Rules for constraints in the user model.....	52
Figure 16: Other rules in the user model	52
Figure 17: Software architecture	57
Figure 18: Part I of the user profile - personal information	58
Figure 19: Part II of the user profile - user's interests and preferences	59
Figure 20: Part III of the user profile - payment privacy (editable mode).....	60
Figure 21: Part III of the user profile - payment privacy (read only mode)	61
Figure 22: Part IV of the user profile – others.....	62
Figure 23: GUI for search query generation.....	64
Figure 24: Presentation window with detailed product information	67
Figure 25: Presentation table containing all the search results	68
Figure 26: Communication through the Model-View-Controller architecture.....	69
Figure 27: The Model-delegate combines View and Controller into UI-delegate.....	69

Figure 28: Shopping cart window	70
---------------------------------------	----

.

List of Tables

Table 1: Roles and examples of agent systems as mediators in E-Commerce	12
Table 2: Techniques used in the sample product recommending agent systems	13
Table 3: Sixteen different types of queries.....	65
Table 4: Test results for different task sets	74
Table 5: Scenarios for testing the shopping cart.....	76

1. Introduction

1.1 Overview of Electronic Commerce

“Electronic commerce”, also known as E-Commerce, refers to a collection of processes that support commercial activities on a computer network. It involves the enablement of a business vision supported by advanced information technology and targets to improve efficiency and effectiveness throughout the trading process. Although the term has gained attention only in recent years, electronic commerce has been in existence in various forms for the past 20 years.

E-Commerce can generally be regarded as a broadening of the term “Electronic Data Interchange” (EDI) which was first introduced in the late 1970s. The airline industry, freight forwarding community and shipping community have all created networks for transmission of EDI and Email data. The growth and acceptance of credit cards, Automated Teller Machines, and telephone banking in the 1980s presented other forms of E-Commerce. While many of these electronic trading technologies had their own effects on markets and created a fair share of publicity, none of them has reached the level that electronic commerce has done in recent years.

The commercialization and privatization of the Internet and its growing popularity are the foundations on which E-Commerce grows. A ubiquitous digital infrastructure that provides an efficient means for communication and information sharing presents an extremely attractive new medium for electronic commerce. For the first time this infrastructure has the potential to deliver what the notion of E-Commerce has indeed implied. In the past, businesses such as banks could conduct activities with each other over closed proprietary networks, but the growth of the Internet has changed this

paradigm, and expanded it dramatically. Traditional E-Commerce activities can now be conducted with millions of new participants on a global scale.

The scope of E-Commerce is substantial, and the numbers make electronic commerce on the Web enticing. In the United States and Canada, approximately 51 million people are using the Internet. Over 73 percent of Web users have searched for information about products and services on the Internet, and nearly 11 percent of Web users have actually purchased online. The interest in electronic commerce is huge and growing. As of March 6, 1998 Yahoo lists 313,163 companies as being accessible through their online search engine, a figure that has been growing by an average of 2,000 to 6,000 every week since the beginning of 1997.

As electronic commerce is growing at a rapid pace, global electronic market will have a profound impact on commerce in future. Most experts predict a radical shift in the way that business will be conducted in the next century. This shift not only has businesses scrambling to meet this new marketing reality, but also raises many important research questions about business strategy, technical infrastructure, security of transactions, user authentication, government policies, the electronic market demographics as well as user's accessibility to this technology.

E-Commerce is essentially a user-centered activity. So the user aspects need to be tackled in all E-Commerce applications. However, we find that existing E-Commerce systems do not emphasize the user aspects to some extent. For example, the following problems have not been solved fully or even not been investigated at all in current electronic commerce applications:

1. Provide personalized service to a particular user by employing a user model
2. Present information in a way that suits the user's needs
3. Provide proactive response and messages to assist the user
4. Disambiguate user's search input based on his user model

The work reported in this thesis concentrates on the user aspect of E-Commerce applications. In this work, we provide one solution to the above problems by using the technology of intelligent user interface agents.

1.2 Agents and User Interface Agents

In order to give a definition of the term “user interface agent”, we need to introduce the concept of “software agent” first. There is no universal definition of the term “software agent”. A generally accepted definition is that of an autonomous software component that performs tasks on behalf of a user. But since this could be said of almost any software program, a software agent should also possess some other desirable properties. Researchers have proposed several properties that could distinguish an agent from conventional software [Wooldridge 95, Mandel 97]:

- *autonomous*: agents are independent entities capable of reasoning on their own. They are able to exercise a non-trivial degree of control over their own actions on behalf of their owner, without requiring explicit permission for every action.
- *reactive*: agents can perceive changes to their dynamic environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined) and respond to those changes with suitable reactions.
- *proactive*: agents are able to exhibit goal-oriented behavior by taking the initiative.
- *social or communicative*: agents can interact with the user or other agents in order to best accomplish their goals.
- *personalized*: agents can either learn or be explicitly taught what to do for each individual or group of users.
- *persistent*: agents involve a sense of temporal continuity, they are usually continuously running.
- *adaptive*: agents track the user’s interests if they change over time.
- *trustworthy*: agents will do what their owners expect them to do.

Software agents make an interesting topic of research in the fields of Human-Computer Interaction and Artificial Intelligence. Now computers are as ubiquitous as cars or televisions, but exploiting their increasing capabilities is still a problem that we need to tackle. As this technology proliferates, the gap between millions of untrained users and a large number of application software dealing with products and services will become even more apparent. Therefore, people have set high hopes on these so-called “software agents”, which “know” users’ interests and can assist end users in several ways.

Agents can assist to reduce the burden of working with large amount of information and data. By adopting intelligent agent technology, users could be engaged in a cooperative process in which both human and software agents initiate communications, monitor events and perform tasks to meet users’ goals. The type of application domains in which agent technologies are currently being applied or investigated include workflow management, network management, air-traffic control, business process re-engineering, data mining, information management, electronic commerce, education, personal digital assistants, scheduling management, etc.

A class of agents that assists users in learning about and effectively using the user interface software has come to be known as “user interface agents” or UIA [Sullivan, 91]. An interface agent has the same characteristics as an intelligent agent but works in a user interface environment. Pertinent information about the user, and details regarding the characteristics of the user interface software and the domain of application are generally stored in a persistent knowledge base of the UIA. The learning ability of the UIA plays an important role in adapting itself to meet its user’s needs. The objective of UIA is to make the software system appear simple and intuitive to use, and finally assist the end users in performing their tasks. This thesis is concerned with user interface agents applicable to E-Commerce applications.

1.3 Overview of the CITR Project

CITR (Canadian Institute for Telecommunication Research) is one of the National Centers of Excellence supported by the Canadian government's research initiatives. There are several major projects managed by CITR and "Enabling Technologies in Electronic Commerce" is one of them in which the author of this thesis participates. The objectives of the project are: (1) developing fundamental enabling technologies for electronic commerce applications; (2) analysis of networks, computer systems and workflow architectures in view of supporting electronic commerce applications; (3) modeling and understanding the behavior and architecture of electronic commerce applications in order to guide the development of the required enabling technologies.

The project consists of following five major components:

- *development of interoperable multimedia virtual catalogs*
This sub-project addresses the multimedia electronic commerce catalog development issues as well as the interoperability of these repositories to enable users to access multiple, distributed and potentially heterogeneous catalogs in a uniform and transparent manner.
- *system and network performance and project management*
This sub-project uses the deployed e-commerce application as a testbed to collect data and conduct measurements on application behavior.
- *quality of service and distributed systems management*
This sub-project addresses the management of quality of service and the adaptation of the application within such system and network environments.
- *user interface and intelligent agents*
This sub-project investigates the design and integration of a user interface employing Distributed Virtual Environment and using appropriate intelligent agent technologies.
- *security issues*
This sub-project considers both the general security issues on the Internet and the specific problems generated by this application.

This challenging project involves 9 professors and more than 20 graduate students from 7 universities in Canada. University of Ottawa and Concordia University are both responsible for the user interface and intelligent agents part. A detailed description of this part will be given in Chapter 4. An additional constraint of this thesis work is to be able to inter-operate with other sub-projects to create a coherent overall project.

1.4 Objective and Scope of the Thesis

The objective of this thesis is to identify the ways in which a software agent at the user interface level can assist the end users in E-Commerce applications. User interface agents can be developed to assist different categories of users in a variety of ways. Studies of the traditional user interface technology have shown the existence of a “gap” between what a user interface software can actually do for the user and the user’s expectations about it. The role of UIA is to bridge or narrow this gap as much as possible.

In order to explore the software agent support in E-Commerce applications, we focus on the following issues in this thesis:

1. Study the users’ shopping behaviors in general, and particularly in E-Commerce applications so that we can employ agent technologies in different stages of user’s shopping experiences.
2. The software agent needs user specific knowledge to assist its users achieving their goals. Therefore, we need to explore the ways of building a user model (profile) to acquire such knowledge.
3. Studies about the user interface require experimentation. Hence we need to develop a prototype as a “proof of concepts” in this thesis.

A user model enables the application system to adapt its performance to each individual user. In our approach, the software agent builds a user model by knowing the user’s

interests, habits, and preferences. This user model will be used to solve the problems (mentioned in section 1.1) in E-Commerce applications. Our user interface agent also has the ability to “reason” effectively with the use of a knowledge base that is stored in the form of inference rules.

Electronic commerce and other information technologies are changing the appearance of global commerce. They can drastically reduce transaction costs, make distance between buyers and sellers irrelevant, and provide access to global markets. Information that was time-consuming and difficult to obtain and store in the past, now can be collected, compiled, analyzed, and delivered around the world more quickly and efficiently than ever before. In fact, nowadays users are often “submerged” into a sea of information and trying hard to meet certain constraints such as time. Therefore, it is important to assist users in filtering information according to his needs, time, resource constraints, and expectations. We deal with this information overload problem by presenting information to the user in a manner that meets his needs and preferences in our prototype of the UIA.

1.5 Organization of the Thesis

Chapter 2 of the thesis provides a literature review of the intelligent agent technologies for E-Commerce. A CBB (Consumer Buying Behavior) model is presented as a common framework for exploring user’s shopping needs as well as the roles of agents in electronic commerce applications. The chapter also introduces several sample agent systems in use as mediators in e-commerce and analyzes the underlying techniques used in these systems.

Chapter 3 first gives a brief introduction to the architecture and roles of UIA. It then addresses the concepts and contents of “user modeling” that are basic components of a UIA. An overview of several typical UIA applications are also described along with sample UIA systems. Finally, we compare these sample UIA systems with our prototype system.

Chapter 4 presents the concepts employed in the CITR sub-project “User Interfaces and Intelligent Agent”. The chapter analyses the users’ shopping behaviors and classifies users into task-oriented groups. It then focuses on the requirements and functions of a UIA prototype for electronic shopping. The user model aspect for prototyping the UIA is explored.

Chapter 5 describes a Java implementation of the UIA prototype. It presents implementation and testing details of various system components.

Chapter 6 summarizes the contribution of the thesis, and provides suggestion for future work.

2. Agent Technologies for E-Commerce

2.1 Users' Shopping Behaviors in E-Commerce

E-Commerce includes a wide range of user-centered issues such as dealing with intermediaries between the producer and consumer, ensuring security in the payment mechanism, providing a good after-sales service, emulating the pleasurable aspects of shopping, and catering to the needs of the wide variety of end users or consumers. Agent technology can be applied to many of these areas. For example, agents can “go shopping” for a user, taking specifications and returning with recommendations of purchases which meet those specifications. They can act as “salespeople” for sellers by providing advice about products or services. Agents can examine a large number of products to help in making a decision to buy or sell. The ability of an agent not only eliminates the need to manually collect information about products but also allows negotiating an optimal price with the various sellers.

It is beneficial to have a common framework for exploring users' needs as well as the roles of agents in e-commerce. The literature is very rich in the field of marketing and consumer behavior. Based on this rich knowledge, Guttman et. al have proposed a model called CBB (Consumer Buying Behavior) which comprises the actions and decisions involved in buying and using goods and services [Guttman 98]. Even though there are multiple models that try to capture the Consumer Buying Behavior (for example, the Nicosia model, the Howard-Sheth model, the Engel-Blackwell model, the Bettman information-processing model, and the Andreasen model), they all share a similar list of six basic stages described below [Guttman 98]:

1. Need identification

This stage characterizes the consumer becoming aware of some unmet need. Within this stage, the consumer can be stimulated through product information.

2. Product Brokering

This stage comprises the retrieval of information to help determine *what* to buy. This encompasses the evaluation of product alternatives based on consumer-provided criteria. The result of this stage is called the "consideration set" of products.

3. Merchant Brokering

This stage combines the "consideration set" from the previous stage with merchant-specific information to help determine *who* to buy from. This includes the evaluation of merchant alternatives based on consumer-selected criteria (e.g. price, warranty, availability, delivery time, reputation etc.)

4. Negotiation

This stage is about *how* to determine the terms of the transaction. Negotiation varies in duration and complexity depending on the market. In traditional retail markets, prices and other aspects of a transaction are often fixed, leaving no room for negotiation. In other markets (such as stocks, automobiles, and fine art), the negotiation of price and other aspects of the deal are integral to the buying process.

5. Purchase and Delivery

The purchase and delivery of a product can either signal the termination of the negotiation stage or occur sometimes afterwards (in either order). In some cases, the available payment options may influence product and merchant brokering.

6. Service and Evaluation

This post-purchase stage involves product service, customer service, and an evaluation of the satisfaction of the overall buying experience and decision. The nature of this stage depends upon for whom the product was purchased.

As with most models, these stages represent an approximation and simplification of complex behaviors. Although CBB covers many areas, it is important to recognize its limitations [Maes 98]. For example, CBB focuses primarily on retail stages (although most of its concepts pertain to business-to-business and consumer-to-consumer markets). Even within retail, not all shopping behaviors are captured (e.g. “impulsive shopping”). Moreover, some issues of e-commerce are beyond the scope of the CBB model such as back-office-management, supply chain management, and other merchant issues.

2.2 Agents as Mediators in E-Commerce

The six stages in the CBB model also elucidate where agent technologies apply to the consumer shopping experience and allow us to more formally categorize existing agent-mediated electronic commerce systems [Maes 98]. From the CBB perspective, we see agents have been used, thus far, mainly in three primary CBB stages: product brokering, merchant brokering, and negotiation. Agents are mainly being used to help customers locate, compare and buy products and services in business-to-consumer retail market, and to negotiate and act on their behalf in consumer-to-consumer markets.

2.2.1 Product Brokering

The *Product Brokering* stage of the CBB model is where consumers determine what to buy. This occurs after a need has been identified and is achieved through a critical evaluation of retrieved product information. Table 1 shows several agent systems that lower consumers’ search costs when deciding which products best meet his personal criteria: PersonaLogic, Firefly, and Tete-a-Tete.

PersonaLogic [URL1 99] is a tool that enables consumers to narrow down the products that best meet their needs by guiding them through a large product feature space. The system filters out unwanted products within a given domain by allowing shoppers to

specify constraints on a product's features. This kind of *constraint-based filtering* uses attributes of items to determine their relevance. However, unlike most feature-based techniques which access data in their native formats, constraint-based techniques require that the problem and solution space be formulated in terms of variables, domains, and constraints. In PersonaLogic, a constraint satisfaction engine returns a list of products that satisfy all of the shopper's hard constraints in the order of how well they satisfy the shopper's soft constraints .

	Persona Logic	Firefly	Bargain Finder	Jango	Tete-a-Tete
Need identification					
Product brokering	√	√			√
Merchant brokering			√	√	√
Negotiation					√
Payment and delivery					
Service and evaluation					

Table 1: Roles and examples of agent systems as mediators in E-Commerce
(adapted from [Guttman 98])

Like PersonaLogic, Firefly [URL2 99] services help consumers find products. However, instead of filtering products based on features, Firefly recommends products via a "word of mouth" recommendation system called *collaborative filtering*. The system first compared a shopper's product ratings with those of other shoppers. After identifying the shopper's "nearest neighbors" (users with similar tastes), the system recommends the products the neighbors rated highly but which the shopper may not yet have rated, possibly resulting in serendipitous finds. Firefly uses the opinions of like-minded people to offer recommendations of such commodity products as music and books, as well as more-difficult-to-characterize products, such as Web pages and restaurants.

In addition to constraint-based and collaborative filtering techniques, two other techniques are widely used to implement product brokering and product-

recommendation agents. A large set of sites uses simple rule-based techniques to personalize product offerings for individual customers. A few sites experiment with data-mining techniques to discover patterns in customer purchasing behavior, exploiting these patterns to help customers find other products that meet their needs.

	Persona Logic	Firefly	Bargain Finder	Jango	Tete-a-Tete
Content-base filtering			√	√	
Collaborative-based		√			
Constraint-based filtering	√				√

Table 2: Techniques used in the sample product recommending agent systems

2.2.2 Merchant Brokering

Whereas the Product Brokering stage compares product alternatives, the *Merchant Brokering* stage compares merchant alternatives. Andersen Consulting's **BargainFinder** [URL3 99] was the first shopping agent for on-line price comparisons. Given a specific product, BargainFinder looks up its price from at least nine different merchant Web sites using Web-browser-like requests. Although a limited proof-of-concept system, BargainFinder offers valuable insight into the issues involved in price comparisons in the online world. For example, one third of the online CD merchants accessed by BargainFinder blocked all of its price requests. One reason was that many merchants do not want to compete on price alone. Value-added-services offered on merchants' Web sites were being bypassed by BargainFinder and therefore not likely considered in the consumer's buying decision.

Jango [URL4 99] can be viewed as an advanced BargainFinder. The original Jango version "solved" the merchant blocking issue by having the product requests originate from each consumer's Web browser instead of from a central site as in BargainFinder. This way, requests to merchants from a Jango-augmented Web browser appeared as requests from "real" customers. Such aggressive interoperability makes it convenient

for consumers to compare prices from a number of merchants' online catalogs, whether or not merchants welcome such comparisons.

Systems like BargainFinder and Jango try to collect information from many different Web sources using *content-based* filtering. Different sources have different inputs (e.g., CGI scripts, Java applets) and presentation methods, so the product recommender systems have to adjust their interaction methods depending upon the Web site. Since there is no standard way of defining and accessing merchant offerings, most recommender systems employ "wrappers" to transform the information from a specific Web site into a locally common format. Different systems follow alternate approaches to creating wrappers. In BargainFinder, the Internet locations of online CD stores and the methods to access them are hand-coded by Andersen Consulting programmers. This method worked well at the beginning but is very hard to scale since it involves updating the wrapper for each site whenever the access methods or catalog presentation format are altered. Jango helps automate the creation of wrappers for new sites by generalizing from example query responses to online merchant databases. This technique is not perfect, but boasts a nearly 50% success rate in navigating random Internet resources.

2.2.3 Negotiation

From the CBB perspective, the Negotiation stage is where the price or other terms of the transaction are determined. The benefit of dynamically negotiating a price for a product instead of fixing it is that it relieves the merchant from needing to determine the value of the good a priori. Rather, this burden is pushed into the marketplace itself. Table 1 shows one representative agent systems that assist customers in negotiating the terms of a transaction: Tete-a-Tete.

Tete-a-Tete [URL5 99] provides a unique negotiation approach to retail sales. Unlike most other online negotiation systems which competitively negotiate over price, Tete-a-Tete agents cooperatively negotiate across multiple attributes of a transaction, for examples, warranties, delivery times, service contracts, return policies, loan options, gift

services, and other merchant value-added services. This negotiation takes the form of multi-agent, bilateral bargaining with an argumentative style and uses the evaluation constraints captured during the Product Brokering and Merchant Brokering stages as dimensions of a multi-attribute utility. This utility is used by a consumer's shopping agent to rank merchant offerings based on how well they satisfy the consumer's preferences.

2.3 Future Directions

Today's online businesses are faced with increasing competition for people's time and attention. To be successful, electronic shopping systems need to provide exceptional value through personalized service to its customers using intelligent agents and good user interfaces. They must be able to deliver a unique service based on their customers' needs. We believe that intelligent agents can help to create that kind of personalized service. These services may come in different forms such as delivering information to the right customers at the right time, helping like-minded customers to meet or communicate with each other, personalized product recommendations, sales promotions tailored to individuals, bargaining the price of a product or service and dialog-based human computer interaction.

Software agents can also help buyers and sellers to cope with information overload and expedite specific stages of the online buying process. Today's first-generation agent-mediated e-commerce systems are already creating new markets such as low-cost consumer-to-consumer goods and beginning to reduce transaction costs in a variety of business processes [Maes 98]. The industries currently using e-commerce are those dealing with perishables (such as travel, theater and concert tickets, and network bandwidth availability), and surplus inventory and commodities (such as gas, electricity, pencils, music, and books).

However, we still have many open issues to explore before software agents transform the way people do business. This change will occur as agent technologies mature to better manage ambiguous content, personalized management of preferences, complex

goals, and changing environments. The greatest changes may occur when standards are adopted to unambiguously and universally define goods and services, consumer and merchant profiles, value-added services, secure payment mechanisms, and inter-business electronic forms.

Looking even further into the future, agents will explore new types of transactions in the form of dynamic relationships among previously unknown parties. Agents will strategically form and reform coalitions in order to bid on contracts and leverage economies of scale. Agents can also create dynamic business partnerships in the sense that the partnerships exist only as long as necessary and then drop. In this new generation of agent-mediated e-commerce, companies will be at their most agile state.

3. User Interface Agents and Their Applications

User Interface Agent (UIA) is a relatively new area of research within artificial intelligence (AI), human-computer interaction (HCI), and user modeling communities. If we have to choose two words to characterize the research in UIA, the words would be “delegation” and “customization”. The AI community as a whole has been concerned with what a UIA can do for the user, whereas the HCI community has been concerned with what the user can do with the agent. The strength of AI research lies in its years of progress in knowledge representation, reasoning, and machine learning, whereas the strength of HCI research in UIA is in its attentiveness to the user needs in performing his tasks.

3.1 Architecture and Roles of UIA

An interface agent has the same characteristics as an intelligent autonomous agent but works in a user interface environment and helps the user with certain computer-based tasks. Its goal is to offer assistance to the user and automate as many of the actions of the user as possible. It may also learn new tasks and predicts actions. The architecture of an interface agent with learning capability is composed of several modules: user-agent interface, domain knowledge base and a learning algorithm. Each of these components is crucial to the operation of the agent and affects its performance. The agent monitors the user’s actions, uses its knowledge to learn about the current situation, and tries to reason about how to assist the user.

A learning interface agent acquires its competence from four different sources as shown in Figure 1 [Maes 94]:

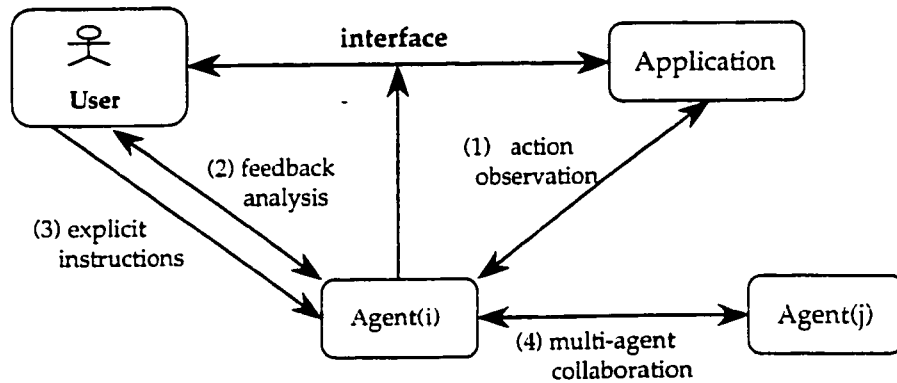


Figure 1: Four knowledge acquisition sources for learning interface agents

1. First of all, the interface agent learns by continuously “looking over the shoulder” of the user as the user is performing actions. The interface agent can monitor the activities of the user, keep track of all of his actions over long periods of time (weeks or months), find regularities and recurrent patterns, and offer to automate these.
2. Agent can learn from the direct and indirect user feedback. Indirect feedback happens when the user neglects the suggestion of the agent and takes a different action instead. The user can also give explicit negative feedback for actions automated by the agent.
3. Agent can also learn from examples given explicitly by the user. The user can train the agent by giving it hypothetical examples of events and situations and telling the agent what to do in those cases. The interface agent records the actions, tracks relationships among objects, and changes its example base to incorporate the example.
4. Another method used by the interface agent to acquire competence is to ask for advice from agents that assist other users with the same task (and that may have built up more experience).

Using various learning mechanisms [Maes 94], interface agents offer the promise of customizing the user interface of a computer system for a particular user. An UIA usually fulfills one or more of the following roles:

- (a) Assisting the user in communicating his task to the rest of the system.

This typically involves presenting the user with an easy to use interface which hides from him the actual underlying system that may be very complex. This should also provide benefits to system developers, allowing them to easily increase the functionality of the existing system by simply slotting in another functional component.

- (b) Learning the user profile (user model)

The underlying software application interacts with the user via the interface (Figure1). The user profile can be incrementally constructed by observing the user's behavior in terms of interactions with the interface. The agent's knowledge of the semantics attached to the individual user actions and the context will help in building the user profile.

- (c) Selecting appropriate presentation of the system output

This presentation should be consistent with the user profile and user's current interactions with the application. The agent is also responsible for presenting the output to the user in a timely and appropriate manner.

3.2 User Modeling

One of the principal problems in constructing an UIA is gathering information regarding the user's interests, goals and general preferences. A user model is an explicit representation of properties of a particular user, which allows the system to adapt diverse aspects of its performance to individual user. Researchers from the fields of AI, HCI, psychology, education, and other areas have all investigated ways to construct, maintain, and exploit user models. Employment of user models increases the effectiveness and usability of systems in a wide variety of situations.

3.2.1 Contents of User Model

The user characteristics reflected in a user model can be separated into application dependent and application independent categories.

Among the application dependent characteristics are [Morris 87]:

- prior experience with computers
- knowledge of the present system and application
- goals, intentions, and expectations

Among the application independent characteristics are [Browne 90]:

- preferences
- psycho-motor skills
- capabilities
- cognitive and learning abilities
- understanding
- motivation

3.2.2 Classification of User Models

There are three different criteria to distinguish among the various user models [Dieterich 93]. One criterion is *granularity*. The scale ranges from a single model for all users, a so-called *canonical* user model, to models for each individual user. In between, there are models for groups of users that have some characteristics in common. Using a canonical model is simplest because it allows the designer to complete the model during implementation. However, canonical user models are of limited value in systems that are used by a heterogeneous user community. The other extreme, *a model for each user*, may be expensive. For this reason often the *stereotypical* approach is chosen. A stereotype is a description of a class of users to specify selected important aspects. Usually a stereotype does not describe all aspects of the user's characteristics, but is restricted to a subset of them. Stereotypes play a major role in building user models,

particularly in the case where the application acquires user information in clusters. The stereotype used in different applications may range from very general ones, which might be appropriate in a wide variety of domains, to very specific ones, which are appropriate only for specific systems and domains.

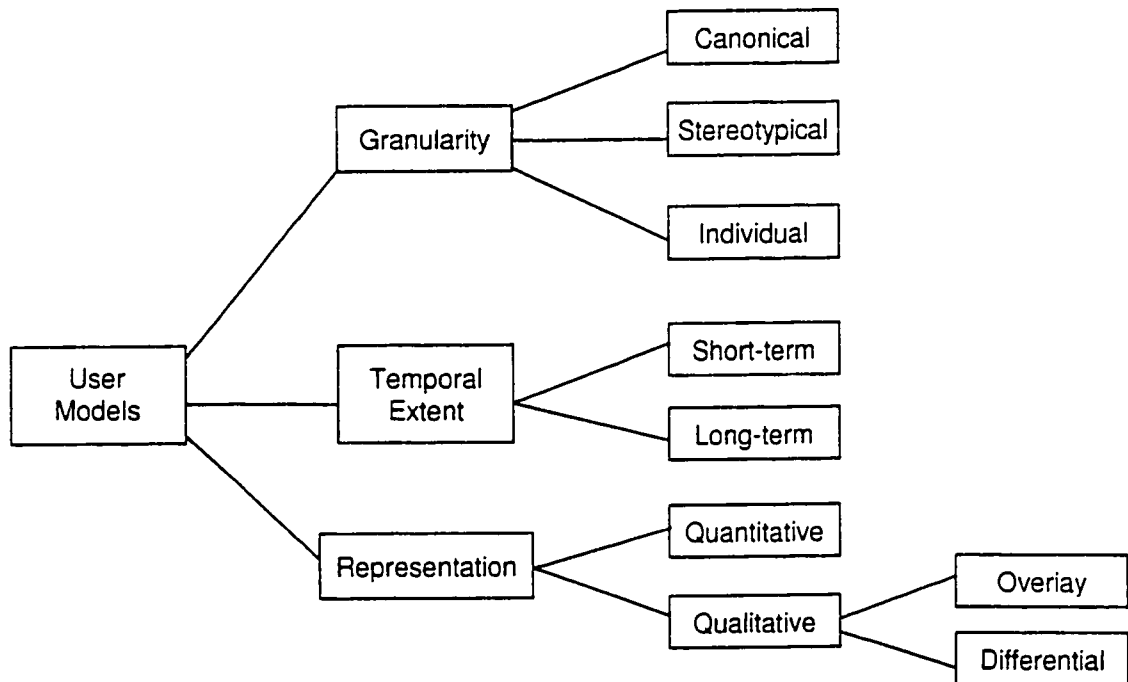


Figure 2: Dimension of user models [Dieterich 93]

Another criterion is *temporal extent* of the data acquired during a session. On the one hand, there are facts which are valid only in the current context or session ("short-term data"), for instance references to earlier situations in the present dialog, the last sentence typed in, or the user's topical level of attention and concentration. On the other hand, there is information which should be kept beyond the current session ("long-term data") and, therefore, be saved on a permanent storage medium. The latter category includes aspects like user's preferences, experience, and capabilities.

Knowledge about the user can be represented by a *quantitative* or *qualitative* model. The first type of user model contains quantitative data like the user's error rate, the number of help requests, or the amount of time which the user has spent with the system. Within the qualitative category one can distinguish between *overlay* models which

encode the user's knowledge as a subset of the knowledge of an expert, and *differential models* which contain the differences of the user's knowledge from that built into the system.

3.2.3 Building and Updating User Model

The actual model can be built in different ways. One can start with an *empty* model, try to classify the user at the very first session into one or more *stereotypical* categories; or build an initial *individual* model based on a preliminary question-answering session. During the interaction with the user these can be refined in the light of further information.

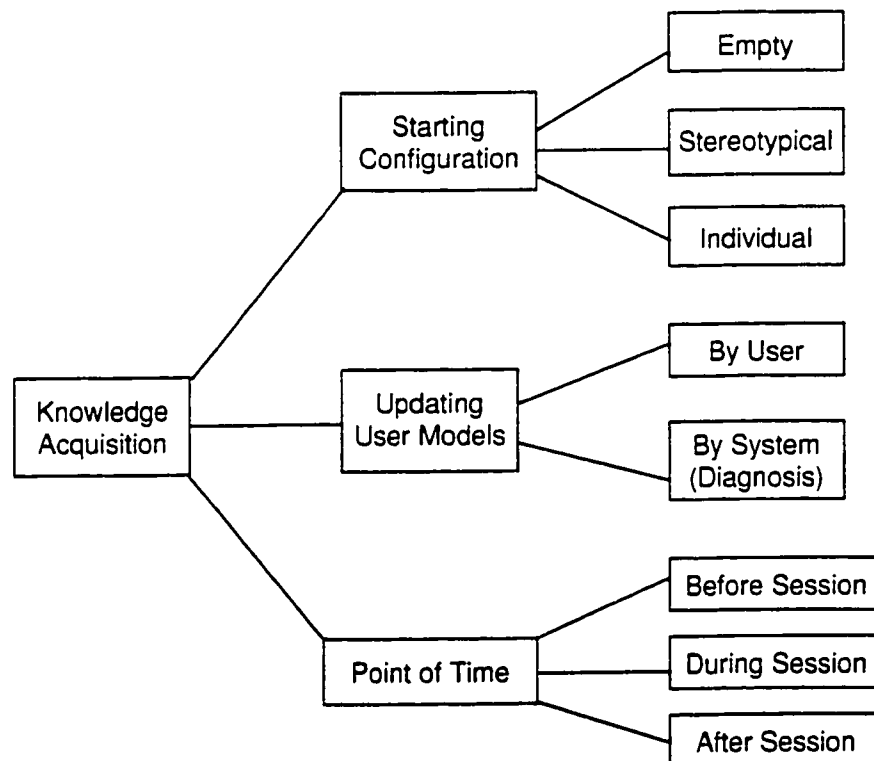


Figure 3: Knowledge acquisition (adapted from [Dieterich 93])

There are two possible ways that the user model can be incrementally updated, either during use of the system or between sessions. One possibility is to let the *users* create models of themselves. This approach is quite simple provided the system is equipped

with the necessary flexibility and the users are capable of doing it. This, however, puts an additional burden onto the user. In particular, novices who do not know what can be modified and how this can be achieved may be overcharged by this task. The second possibility is for the *system* to incrementally collect facts about the users. This knowledge acquisition can be achieved either directly by questioning the users at the beginning of a session or the first time they interact with a given system, or indirectly by monitoring the user interaction with the system during the session.

3.3 Application of Intelligent Interface Agents

The main application areas for UIA are those in which the knowledge and reasoning capability of an agent can meaningfully supplement the user's interactions with the application software. The user's request may be incomplete or vague to meet his real needs. The UIA can supplement the missing information or refine the request to help the user achieve his goals. Three typical application areas that can be characterized this way are information filtering, information retrieval, and intelligent assistant.

(1) Information Filtering Agents

In open networks such as the Internet, it is comparatively "cheap" to distribute information to a very large group of recipients. For recipients, this means that they are flooded with large amount of information and faced with the tough task to extract the information that is really relevant or interesting to them. Information filtering techniques aim to provide efficient means that can be used to aid users in navigating through large and diverse information space and selecting the information that is relevant.

The filtering agent is probably one of the simplest classes of intelligent interface agents. This class of system typically consists of a number of large information sources that are to be filtered in some way to leave only the information relevant to the user. Two major approaches to the construction of such systems exist, those which utilize explicit user models and those which do not. Of those systems which do not utilize explicit user

models the most effective technique by far is “Collaborative Filtering” which was mentioned in section 2.2.1.

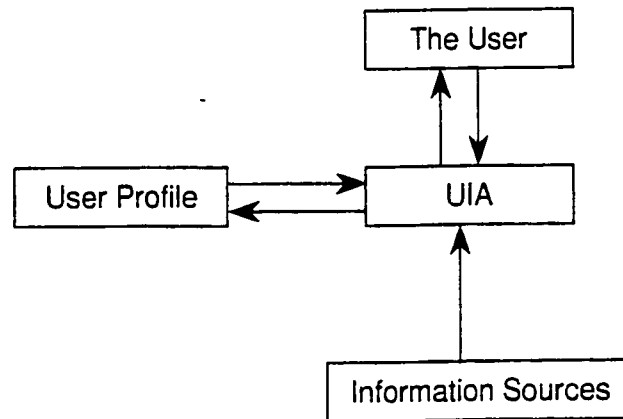


Figure 4: Reference architecture for information filtering agents

(2) Information Retrieval Agents

Information retrieval systems deal with relatively more stable collection of information items. Digital libraries are good examples of such systems. Different categories of users access the information retrieval systems for widely diverse needs. When a query is not specific enough, a very large amount of items are retrieved from the database or other information sources. So a traditional information retrieval system is a good candidate to perform information filtering. There are a number of systems which performs an information filtering role, some of which filter in an autonomous way, presenting the user with the information the system considers to be of interest to its user. Similarly, this same type of system can also be proactive when it actively searches for the information that it judges would be of interest to the user.

Although the architecture diagram of the information retrieval agents (Figure 5) may appear at first glance to be rather similar to that of the information filtering agents (Figure 4), the inclusion of the arrow indicating a flow of communication from the interface agent to the information resources reflects a significant improvement in sophistication. This is the class of system that exhibits a significant degree of proactivity. The extra arrow on the reference architecture indicates that the interface agent requests particular items of information from the information sources, rather than passively filtering information presented to it. It is clearly of importance that the

information requested satisfies the users' queries or goals. Thus, the request sent by the interface agent must be based both on the information obtained directly from the interface and also upon any user models the system has managed to build up.

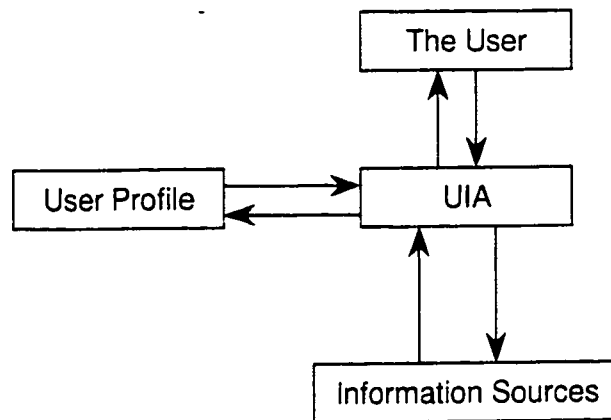


Figure 5: Reference architecture for information retrieval agents

(3) Intelligent Assistant

This reference architecture shown in Figure 6 embodies what is probably the dominant type of interface agent. The intelligent assistant attempts to help its users perform their routine day-to-day tasks. This type of system would ideally have domain knowledge which is the knowledge of the capabilities of the system it is assisting the user to use, as well as a model of the user's abilities and preferences. The agent is then responsible for providing help or advice based upon these knowledge sources and the user's current task as perceived by the agent.

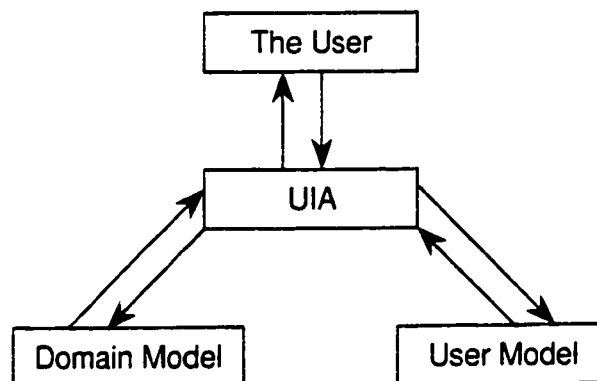


Figure 6: Reference architecture for intelligent assistants

3.3.1 Case Study 1: A UIA for Electronic Mail Handling

Electronic mail filtering is a very popular application domain for UIA. One example of this type of system is found in the work of Pattie Maes at MIT Media Lab.

3.3.1.1 *System Overview*

Maxims [Lashkari 94] developed at MIT Media Lab is a learning interface agent which assists users with electronic mail handling. Maxims monitors the actions of the user when he deals with his emails, finds recurrent patterns and offers to automate them. For example, if an agent notices that a user almost always stores messages sent to the mailing-list “intelligent-agent” in the folder AI Mailing Lists, then it can offer to automate this action next time a message sent to that mailing list is encountered. The agent can also automate reading, printing, deleting, replying and forwarding emails as well as assign priority to messages.

3.3.1.2 *Learning user patterns*

The algorithm that attempts to capture user patterns for Maxims is called *Memory-Based Reasoning*. The implementation is based upon the concepts of situations and actions. In the electronic mail domain, mail messages along with some context information are chosen to represent situations, and the user’s handling of the messages forms actions. When the user takes an action, it is paired with the corresponding situation and the situation-action pair is recorded in the agent’s memory. For example, if the user reads a message M, the pair $\langle M', \text{read-action} \rangle$ is memorized, where M' contains details about the message M and relevant context information.

When a new situation occurs, the agent will try to predict the action of the user, based on the examples stored in its memory. The agent compares the new situation with the memorized situations and tries to find a set of nearest neighbors (or close matches) to the current situation. The memorized situation with highest similarity contributes to the decision of which action to take or to suggest in the current situation. The distance metric used is a weighted sum of the differences for the features that make up a

situation. Some features carry more weights than others. The weight of a feature is determined by the agent. Occasionally (e.g., at nights), the agent analyzes its memory and determines the correlation between features and actions taken. For example, if the agent detects that the “from” field of an email message is highly correlated to the receiver reading the message, while the “date” field is not correlated, the detected correlation is then used to assign the weights in the distance metric.

After predicting an action for the user, the agent must decide how to use that prediction. For each possible action, the user can set two confidence thresholds: the *tell-me* threshold and the *do-it* threshold. If the confidence in a prediction is above the tell-me threshold, the agent displays the suggestion in the message summary line. If the confidence is above the do-it threshold, the agent autonomously takes the action. The agent’s confidence in its predictions grows with experience, which gives the user time to learn to trust the agent.

An agent starts out with no experience. As messages arrive and the user takes action, its memory grows. Only after a sufficient number of situation-action pairs have been generated, the agent is able to start predicting patterns of behavior confidently and reasonably accurately. In order to deal with this slow start problem, two additional competence acquisition schemes are proposed.

First, it is possible for the user to instruct the agent explicitly. If the user does not want to wait for the agent to pick up a certain pattern, then the user can create a hypothetical situation and show the agent what should be done. This functionality is implemented by adding the example to the agent’s memory, including “wildcards” for the features which were not specified in the hypothetical situation.

A second method is based on multi-agent collaboration. When the agent does not have enough confidence in its prediction (confidence lower than “tell-me” threshold), it asks for help from other agents that are assisting other users with electronic mail. The agent sends part of the description of the situation to other agents via email and awaits their response.

3.3.2 Case Study 2: IR-NLI II — A UIA for Intelligent Information Retrieval

IR-NLI (Information Retrieval – Natural Language Interface) II is an intelligent user interface system that allows end users to access online information retrieval systems [Brajnik 87]. It encompasses user modeling capabilities to adapt its behavior to different types of users. The construction and refinement of the user models rely heavily on the basis of stereotypes, which represent typical classes of users.

3.3.2.1 *System Design*

The architecture of IR-NLI II is shown in Figure 7. Two major subsystems can be identified: the UIA and the user modeling subsystem. The UIA is designed to conduct the search session, interact with the user, and interrogate the information retrieval system.

The user modeling subsystem, on the other hand, is designed to carry out the user modeling activity, both within a search session and over several sessions. It performs two basic tasks: (1) extraction of information relevant to user modeling from the dialogue between the user and the system and (2) construction and updating of the user model.

The two communication channels between the UIA and the user modeling allow, respectively, the transfer of information regarding the current user (flowing from the UIA to the user modeling) and request for further knowledge issued by the user modeling subsystem. Both subsystems can work in parallel, synchronizing just for information exchange.

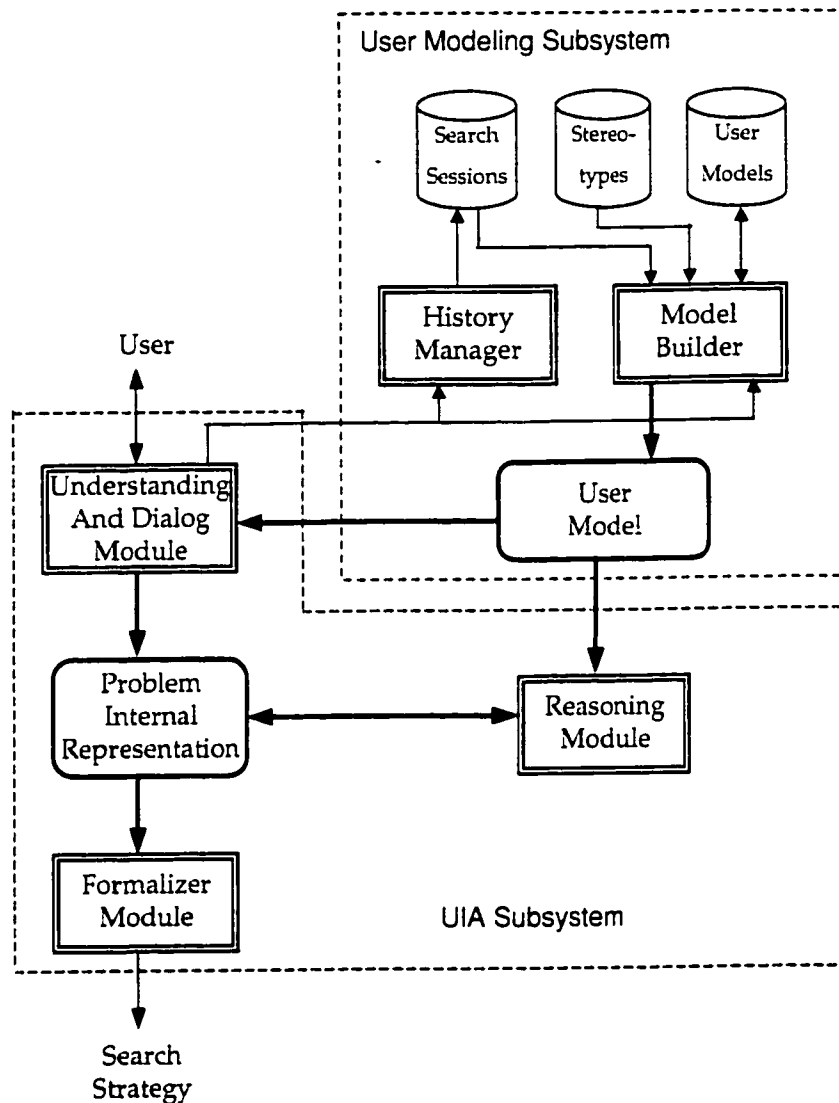


Figure 7: Architecture of IR-NLI II (adapted from [Brajnik 87])

The UIA comprises three modules:

1. The understanding and dialogue module, which manages the interaction with the user, including input comprehension and question generation.
2. The reasoning module, which analyzes, incrementally refines, and completes the initial description of the information problem given by the user and produces a search formulation suitable for constructing the appropriate search strategy.
3. The formalizer module, which constructs the search strategy and actually connects with the information retrieval system.

The working memory of the UIA contains the information provided by the user or acquired from the user model that describes the current information problem.

The user modeling subsystem is composed of two modules:

1. The model builder, which constructs and updates the user model.
2. The history manager, which records a summary of each search session into a long-term data base.

3.3.2.2 Content and Usage of the User Model

At the conceptual level, the user model is representable by a frame structure divided into two parts: user profile and user knowledge. The user profile subframe encompasses knowledge about specific features, attitudes, and traits of an individual user. The user knowledge subframe is devoted to storing information describing what the user knows about the environments of IR-NLI II operation. In addition to these two subframes, a user model includes a user identification name, and a model history, which contains information about the history of the model such as creation date and successive refinements and updating. The resulting structure of the user model is therefore:

```
MODEL < user identification name >  
      < model history >  
      < user profile >  
      < user knowledge >
```

As seen from the architecture of IR-NLI II, the user model can be used by the reasoning module and by the understanding and dialogue module. In general, it supplies further information that supports the system when performing some operations, the most important of which are: (1) tuning the user-system dialogue. (2) translating user utterances and answers to suitable queries for database searches. (3) extracting relevant information from the user model when that information item can not be acquired from the user.

3.3.2.3 Tasks of the Model Builder

The model builder starts by identifying the user or user category accessing IR-NLI II. After this preliminary phase, the model builder looks in the user models base for the individual model: If the user is new, no model exists and model building has to start from scratch. Later, the operation of the model builder proceeds through the five phases described below:

1. *Preliminary interview.* This phase is devoted to acquiring basic information about the user, through a bounded scope, system-driven dialogue.
2. *Stereotype activation.* The preliminary user information gathered in the previous phase is used here to test the activation methods of the stereotypes available in the stereotypes base. All stereotypes whose activation method is satisfied become active stereotypes and are candidates for constructing the individual model of the user.
3. *Stereotype discrimination.* This phase considers the set of active stereotypes and aims at identifying the one that will be used as the kernel of the model construction.
4. *Model refinement.* This phase is aimed at incrementally extending, tuning, and refining the individual model of the current user during a single search session.
5. *Closing operations.* At the end of the search session, the individual model of the current user is stored in the user models base. Moreover, a summary of the current search session is stored in the search session history.

The task of the model builder is only slightly different when the current user is already known to the system and therefore an individual model is available in the user model base. In such a case, phases 1, 2 and 3 above become meaningless and are substituted by two new ones directed at initializing the activity of the model builder before model refinement is started. Thus, the operation of the model builder encompasses the following four phases:

1. *Model retrieval.* The model of the current user is retrieved for further processing.
2. *Historical information processing.* This phase is concerned with statistical processing the summaries of the past search in order to refine typical values of some slots of the model.
3. *Model refinement.* See phase 4 above

4. *Closing operations.* See phase 5 above

3.3.3 Case Study 3: User Modeling in A Commercial Software System TIMS

Modern software systems are complex, and they often support a wide variety of tasks and diverse groups of users with differing problems and needs. This has led to an increased focus on the user in commercial software development and an increasing research focus on user modeling and user adapted interaction.

Despite the many details of design, implementation, maintenance, and performance overhead, there is a much broader reason for the lack of commercial user modeling systems that has been speculated upon in the literature. Many of the approaches embodied in research systems are too complex or impractical for use in commercial software systems. The identification of this shortcoming has prompted researchers to create pragmatic user modeling architectures and techniques, simplified from theoretically-motivated approaches. From a commercial standpoint, this “good enough” user modeling involves the inclusion of a minimalist user modeling components that have some of the major advantages demonstrated by large search systems, but associated with minimal cost for commercial applications.

3.3.3.1 *Introduction to TIMS*

The Tax and Investment Management Strategizer (TIMS) is a sophisticated commercial software system for financial planning which allows a financial planner to analyze the financial status of a client and to demonstrate and evaluate various financial planning strategies in order to improve the client’s financial situation [Strachan 97].

Much of the user’s interaction with TIMS is performed through system components known as *Strategies*. Strategies implement financial planning techniques that human planners suggest to their clients or those that clients already have in place to properly reflect their overall financial picture. Like other systems functioning in complex domains, TIMS must support a diverse range of users: many financial planners have

little computer expertise, some are sophisticated computer users with little or no need of lengthy demonstrations and extensive explanation facilities; many TIMS users are expert financial planners with extensive knowledge of the domain and use TIMS mainly for convenience, others have only limited financial planning skills and will rely on TIMS to fill in gaps in their own knowledge.

A study conducted on the first commercial version of TIMS, released in 1995, confirmed the power and sophistication of the system, but identified concerns with the complexity of the system and the impact of this on novice users. These concerns, along with the breadth of user population that must be supported and the goals of TIMS itself, led to the decision to begin developing an adaptive user interface for TIMS using a minimalist user modeling component.

3.3.3.2 Architecture of the User Modeling Component

In the initial stages of developing a user modeling component (UMC) for TIMS, it was decided to focus on the necessary adaptations for novice users. This was primarily because TIMS was originally developed in collaboration with expert financial planners and had already been structured to reflect the organization of material that was helpful to them. A focus on novices, however, does not eliminate expert users from consideration. The usability of the software must not be limited by the adaptations that are necessary to support novices.

The architecture of the UMC consists of a long-term user model (also stored in a user model database), a task model containing descriptions of a subset of tasks in the system such as rankings of the complexity of Strategies, and an application model containing information about the relationships between a subset of the tasks in the system such as information on Strategies or combinations of Strategies that are equivalent to one another). The inference engine contains simple rules about the relationships between these three models and the possible adaptations that can be performed on the application and the interface.

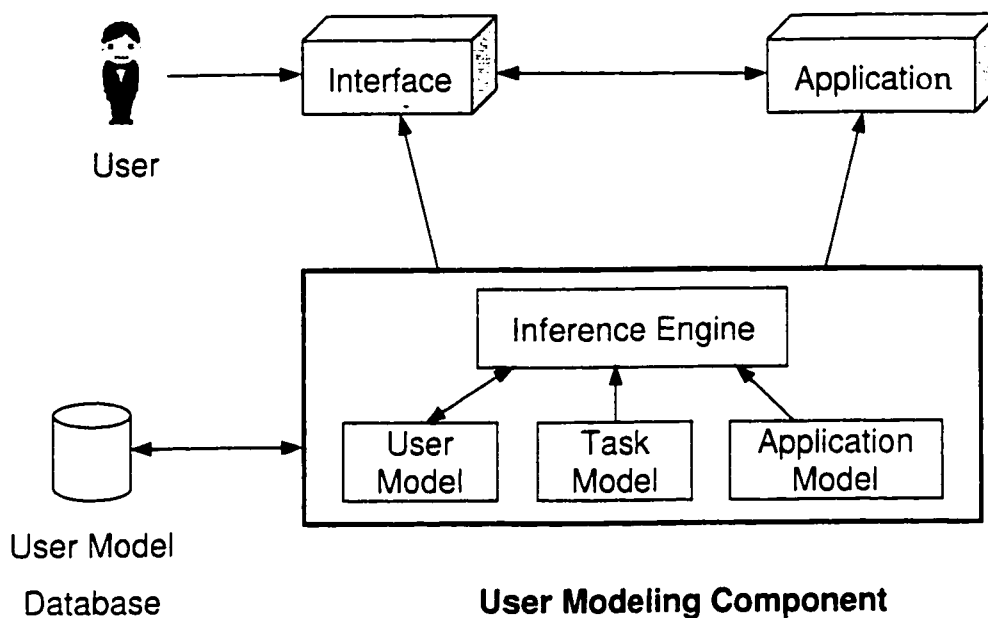


Figure 8: The User Modeling Component architecture of TIMS [Strachan 97]

When the TIMS user first logs on, his user model is ready that is based on a set of stereotypes triggered by his job title and a self-assessment of Windows experience and previous TIMS experience. Because of the diversity of TIMS users, a decision was made to have four stereotypes within TIMS: (1) Novice TIMS users, (2) Novice Financial Planners, (3) Experienced TIMS users, (4) Experienced Financial Planners. The combination of the appropriate Financial Planning and TIMS experience stereotypes results in a basic user model containing parameter settings, specific to the various adaptations that are supported by the UMC. While the most significant updating of the user model occurs during the initialization phase of the system, the user can also update his own user model at any time. Other modifications occur when the tasks represented in the task model are used.

Over time, the adaptive system gradually diminishes into the original, non-adaptive system. Because the UMC is focusing on the support of novice users through the system, support is terminated once the user has achieved a minimum level of proficiency and has used the system for a predetermined period of time. Different types of adaptations in the system have different termination points. The intermediate or

expert user starts with fewer supports, which gradually diminishes as the system has been used for a period of time or once the user has decided to cancel the adaptations. In all cases, the user has ultimate control in terminating the supports.

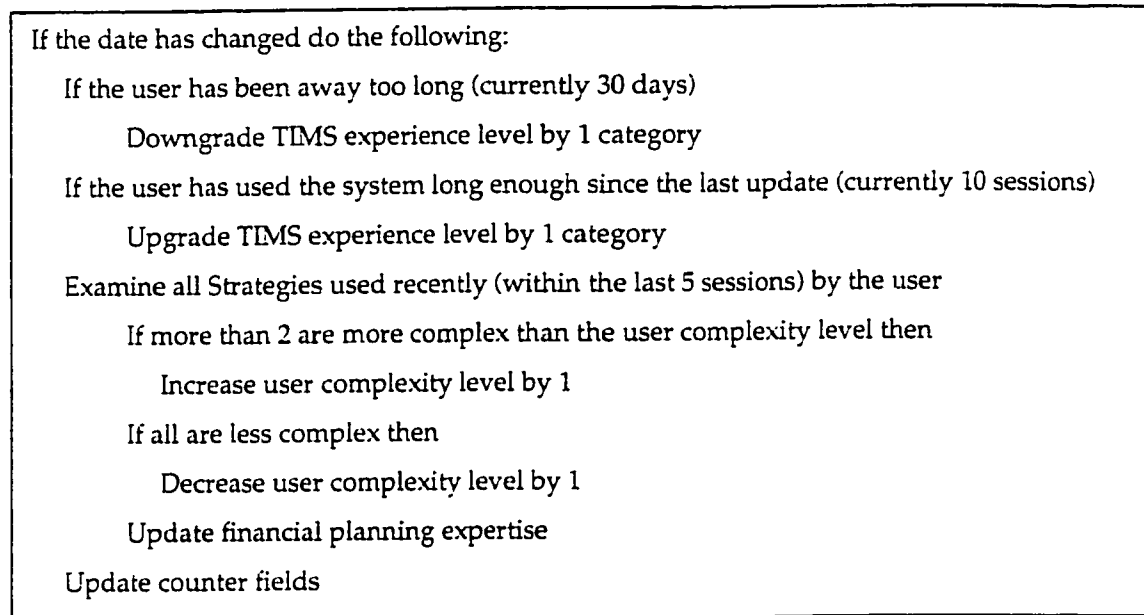


Figure 9: The basic algorithm for updating user model [Strachan 97]

3.3.3.3 Supported Adaptations

The first major phase of adaptation involved adding support for animated demonstrations of the general structure of the system, its components, and the various tasks the user can perform. After viewing a demonstration, the user is able to replay the demonstration and also the user is instructed as to alternate methods of accessing the demonstration in future. Like the other adaptations to be described, the demonstration facility is directly controlled by the UMC. When the user attempts to implement a TIMS Strategy, the UMC is called to determine if a demonstration is to be shown to the user. This adaptation step benefits the novice user most during the first few sessions, and conversely, requires the greatest amount of input from the user in the first few sessions.

The second set of adaptations were designed to recommend simpler alternatives to complex actions in the system. These adaptations rely heavily on the task and

application models maintained in the UMC. All the relevant Strategies are represented in the task model and include a numeric complexity rating. A complexity rating is also maintained as part of the user model, and when a complex procedure is invoked, this rating is compared with the complexity rating of the task in the task model. The application model contains a mapping between complex tasks and low-level tasks, and will be checked for equivalent Strategies if the user's rating is lower than the complexity rating of the task they have selected. The user may choose to disregard any suggestions, and may also choose to deactivate subsequent suggestions at any time.

Like other components of the user model, the user's complexity rating evolves over time as a result of the user's interaction history with the system. The basic algorithm for updating the user complexity level was seen in the initialization routine in Figure 9. The user complexity level is increased by 1 if the user has invoked at least two Strategies that have complexity rating higher than the current user complexity rating, and is decreased by 1 if the user has not recently (within the last five uniquely dated sessions) used any Strategies that are at least as complex as the user complexity rating.

The third phase of the implemented adaptations is intended to address concerns with the initial data entry process. Like many other systems, TIMS gathers data through a series of dialogs that flow in a logical sequence. It is possible to accidentally terminate the series of dialogs by hitting the wrong button on the dialog. This was safe guarded by prompting the novice user and allowing him to return to the data entry loop if it was abnormally terminated. The use of this warning dialog to reduce accidental loss is directly linked to the user model, so that it would not be displayed for experienced TIMS users.

3.3.4 Summary

The UIA has been developed into a practical technology, and put to use in a variety of applications. This chapter has introduced three sample UIA systems that have different architectures and fulfil different tasks. Each of the three examples has its advantages as well as its limitations:

- The interface agent in Maxims requires very little background knowledge since the agent learns the user pattern by observing the way that the user handles his electronic mails. But this learning approach has disadvantages in terms of the computation time and storage space.
- IR-NLI II system is able to recognize the specific characteristics of each individual user by embodying user modeling capabilities. Novice users, experienced users with specific background in the search domain, and expert users are treated in different ways. The authors [Brajnik 87] note that the stereotype management can be further improved by using backtracking and approximate reasoning techniques in order to make the task of the model builder more effective.
- TIMS uses a simple user model. There are very clear advantages for the use of a simple user modeling in complex systems. Users are assigned a single stereotype that is updated as new information is presented. Neither sophisticated learning mechanisms nor uncertainty management schemes are necessary. Nevertheless, the serious drawback of a simple user model over a more complex one is the potential for misjudging users due to the generality of the attributes recorded by the simple model. But the impact of an initially inaccurate user model can be minimized if this aspect is considered in the design of the intelligent UIA systems.

Interface agents radically change the style of human-computer interactions. They can simplify the use of computers, allowing users to move away from the complexity of command languages or the tedium of direct manipulation toward intelligent, agent-guided interactions. The UIAs offer the possibility of providing friendly assistance so that users may not even be aware of the agents' activities. Now interface agents are becoming more and more attractive due to the increasing complexity of user interfaces and the tasks to which they are applied. Even though the results obtained with the current UIA applications are encouraging, many open issues remain, for example, what is the best metaphor for interface agents? What kind of user modeling approach is better for improving the competence of the UIAs?

3.3.5 Comparisons

We compare the UIA systems described in the three case studies with our UIA system for E-Commerce applications in this section. We also present reasons why we chose these sample systems as case studies here.

(1) Maxims system [Lashkari 94]

The learning technique in Maxims requires the application system involve a substantial amount of repetitive behavior within the actions of the user. If this condition is not met, a learning agent will not be able to learn because there is no regularity to learn in the user's actions [Maes 94]. In order to use this learning technique, enough data should be accumulated over a period of time. Our system does not follow this learning approach. Because electronic commerce is a growing new area, and its applications are not as popular as electronic mails at this stage. We need to wait until the interface is used extensively. Eventually the UIA may be incorporated with a learning capability. Which learning methods will be suitable for E-Commerce applications? This is still an open issue.

The UIA in Maxims uses the "situation-action" pairs to model the user behavior. The situation is a "feature set" in electronic mail domain, while the action is performed by the user. As the user performs actions, the UIA memorizes all of the situation-action pairs generated. After gathering the closest matching situations in memory, the agent can calculate a prediction for an action when new situation occurs. In our system the "action set" can also be monitored by the UIA. The user operations include clicking on certain buttons to search the product of interests, to browse the matching products from a database search, to add some items to his shopping cart, to send his order, or to update his user profile. But the "feature set" remains open in electronic commerce applications. Our prototype serves as a starting point for future studies in the selection of "feature set" and "action set". This will be useful for memory-based machine learning.

(2) IR-NLI II system [Brajnik 87]

Although the paper about this system was published in 1987, it addressed the user model aspects very well. This is the reason that we chose it as one of the case studies. Like IR-NLI II system, ours also contains the UIA and the user model. The comprehensive user model in IR-NLI II is based on taxonomy. The user profile in that user model is not as complete as in our system. IR-NLI II also supplements missing information in the user's search query, based on the user model. It uses some "search strategy" (the meaning is not quite clear in the paper). We use JDBC to communicate SQL query with any kind of database system, and we also deal with missing information in the user's query.

IR-NLI II classifies users into two stereotypical categories: generic and expert users. Expert users inherit some properties from generic users. We classify users based on their shopping behaviors. But there is no hierarchical relationship among different user categories. The same user may fit into multiple categories based on his goals and tasks. The UIA in IR-NLI II system interacts with the user through dialogs. The proposed IR-NLI II system was under construction on the LISP platform in 1987, and it is not clear if it was complete. However, we have completed our prototype on the Java platform and tested it.

(3) TIMS system [Strachan 97]

The models in TIMS provide simple solution to a complex task. They deal with financial planning problems. There are three models in this system: user model, task model and application model. The term "Strategy" is used to denote the financial plan of a user. A task model in the specific domain is created using Strategies. An application model captures the relationships between Strategies.

TIMS classifies users into four groups based on their job titles and experiences in using the system. The system focuses on how to adapt its behaviors to different user groups (by providing animations, by recommending simpler alternatives, etc.). Recognizing to which user category that a particular user belongs is one of the major tasks of TIMS system. However, we do not address this issue at all.

Like TIMS system, ours contains an inference engine. The inference engine in TIMS provides adaptations on the application and the user interface, while our inference engine focuses on assisting the user by providing both proactive and reactive responses. Finally, TIMS is a commercial system for financial planning, but ours is a prototype for E-Commerce applications.

4. A UIA Prototype for E-Commerce

4.1 CTR Subproject – User Interface and Intelligent Agent

The user interfaces that most electronic shopping systems provide currently can be viewed as an extended “electronic catalog” to which search capabilities are added. Although the search feature is an improvement over the conventional paper catalog, today’s electronic catalogs are still not good enough to satisfy users’ needs in e-commerce applications. For example, the following issues are not addressed in today’s GUI based user interfaces: (a) Natural and direct dialogs between a user and a vendor. (b) Context or domain based interpretation of the possibly incomplete user specifications for buying. (c) Learning about the user’s habits and preferences and keeping track of persistent information. (d) Providing active consultations with a group of people in whom the user has confidence with respect to the purchase of the goods and services under consideration. Our approach to help overcome these shortcomings is to incorporate software agent support in a virtual shopping mall that simulates most of the actual shopping environment and user interactions. This approach allows us to provide personalized service and customized information to an individual customer.

The objective of the CTR subproject “User Interface and Intelligent Agent” is the design of a user interface for the e-commerce major project using a virtual shopping environment. Users can navigate through the virtual shopping mall where both the customers and sales clerks are represented on the screen by 3D animations, called “avatars”. In this navigation process, users may find some interesting items and add items to their shopping cart. A more sophisticated search for an item of interest, without navigating through the maze of the electronic mall, is that users issue a command to an intelligent agent. The software agent retrieves matching products from which users

select the items of interest and add to the shopping cart. Users are also provided with the flexibility to negotiate the product price with software sales agents or to communicate in real-time with a sales person, if they wish to do so [Georganas 98].

Whereas standalone navigation of a virtual shopping mall may be tolerant to delay, real-time communication with an avatar or a real person requires QoS management. Hence, there is a natural link of this subproject with the QoS Management subproject. On the other hand, purchasing of items either retrieved through virtual mall navigation or through search, we need a secure transaction protocol. Thus, this subproject will also link with the Security subproject.

4.1.1 User Interface Agent for E-Commerce

As mentioned above, instead of navigating through the virtual shopping mall, the user may activate an intelligent user interface agent for retrieving items of interest. The UIA transmits the user's request to a remote multimedia database server and by querying it, returns the matching information. The UIA can assist the user either reactively by responding to user actions, or proactively by monitoring certain events and drawing user's attention if necessary.

A UIA maintains a knowledge base about the user, derived from a user model that includes tasks, preferences, constraints, and the user's shopping characteristics and choices. The UIA is designed to be capable of dealing with user's incomplete or ambiguous query by making use of context based substitution according to the user model. The UIA will also verify with the user about this substitution if the need arises. This thesis is focused on the user model aspect of the UIA.

A UIA observes a user's behavior, and then by applying machine learning techniques, incrementally builds attribute values for the attributes used in a user model. For example, a UIA uses the past behavior of a user to anticipate that user's preferences and interests, e.g., his preferred store to purchase certain products, his usual price range, his favorite manufacturer (brand), etc. For this purpose of anticipation, it becomes

necessary to characterize the “situations” under which observed attributes can contribute to learning and to decide which learning methods are better suited for the selected domain of application. We have not explored this issue because it is beyond the scope of this thesis.

In an endeavor like electronic commerce, it is quite natural to employ multiple agent technology. The term “multiple agents” implies agents of the same user and of different users. The UIA will have an architecture to coordinate multiple agents when they coexist and communicate with each other. For example, there are situations when one user’s agent interacts with agents that assist other users with the same task (that may have accumulated more experiences). An agent may consult another agent about the quality of a product, the reliability of a vendor, or the level of satisfaction of the services provided by a vendor. In this context, inter-agent communication protocol, e.g., KQML (Knowledge Query and Manipulation Language) can be investigated and utilized. A multi-agent communication architecture that enables the UIA to communicate with the software sales agents has been implemented by Patrick Desharnais in the Department of Computer Science, Concordia University [Desharnais2 99].

4.1.2 Software Sales Agent for Price-Negotiations

In physical retail markets, prices are often fixed and not subject to negotiation. But even in these so-called *fixed-price* markets, price negotiations between a vendor and a consumer still take place from time to time [Desharnais1 99]. This phenomenon happens because there are situations when vendors feel it is in their own interest to participate in price negotiations. Even though they might end up selling at a price lower than their fixed price, the expected payoff of doing so is deemed worth it. For example, a vendor might be willing to be flexible in his fixed price policy in order to keep a profitable client satisfied, to make room in the inventory, or to sell an item that has not been selling well.

Currently existing electronic online retail stores do not offer bilateral price negotiation capabilities. From a consumer’s perspective, price negotiation gives them a chance to get better prices. From a vendor’s perspective, the ability to negotiate a price with the

consumer gives them the flexibility to reach a deal which is not otherwise possible with a rigidly fixed price policy. Therefore, it is beneficial to study how software sales agents can be used to negotiate with the customer on behalf of the vendor. A simple and intuitive negotiation protocol is being developed to explore agent support in the negotiation stage of the CBB model by Patrick Desharnais [Desharnais2 99]. For merchants to be interested in negotiation, the negotiation process should be a win-win type situation where both the vendor and the customer are satisfied.

4.2 Users' Shopping Behaviors

In order to provide personalized services to customers in the six stages of the CBB model, we have analyzed the users' shopping behaviors and categorized them into the following three task-oriented groups:

I. Comparative Shopping

The user has a clear idea of what he needs to buy; he compares the prices, stores, warranties, etc. with regard to his shopping preferences.

II. Planned Shopping

The user has a specific product or product category in mind; he wants to get more information of this product or similar products and plans to buy it in the future.

III. Browsing-Based Shopping

The user has no specific item in mind; he wants to get some product information but has no intention to buy.

The time spectrum can be divided into "for now" and "for future", according to a fuzzy boundary [Desharnais1 99]. This division distinguishes between *comparative shopping* and *planned shopping*. The former is "for now" and the latter is "for future". In the case of *browsing-based* behavior, the stage 1 of the CBB model is applicable because the need

for shopping is not yet clear to the user. The *comparative shopping* behavior is relevant to stage 3 and stage 4 of the CBB model as the user has a clear idea of what (or what kind of things) to buy and the user wants to buy now. *Planned shopping* spawns both stage 2 and stage 3 of the CBB model, as the user has still time to select the product and is not yet ready to negotiate.

The three groups we defined can overlap and are by no means disjoint. For example, *browsing* can turn into *planned shopping*, and with the progression of time or change of conditions, *planned shopping* can turn into *comparative shopping*. The detailed requirements for *browsing*, *planned* and *comparative shopping* would be different. Some requirements would be common to more than one type of behavior, such as easy access to pertinent information about the product or service. A set of sample requirements to be fulfilled by a supporting system for these three shopping behaviors is given below:

User Requirements for Comparative shopping:

- R.1.1 Ability to find information about selected vendors
- R.1.2 Assistance in selecting a trade-off between quality and price
- R.1.3 Easy way to specify search constraints (in an incremental way)
- R.1.4 Ability to negotiate with the vendors

User Requirements for Planned shopping:

- R.2.1 Ability to consult other software agents or friends
- R.2.2 Proactive notification of related or new products/events (including reminders)
- R.2.3 Support for product and merchant brokering
- R.2.4 Ability to launch a "monitoring-agent" that will keep looking for things and then report

User Requirements for Browsing based shopping:

- R.3.1 High bandwidth (fast) transfer between servers and clients for quick response
- R.3.2 Ability to display multiple catalog items at a time on the screen
- R.3.3 Ability to view other shoppers' browsing through items (naturalness)
- R.3.4 Faithful and realistic display of the products (color, size etc.)

4.3 Prototyping the UIA

4.3.1 Objective of the UIA Prototype

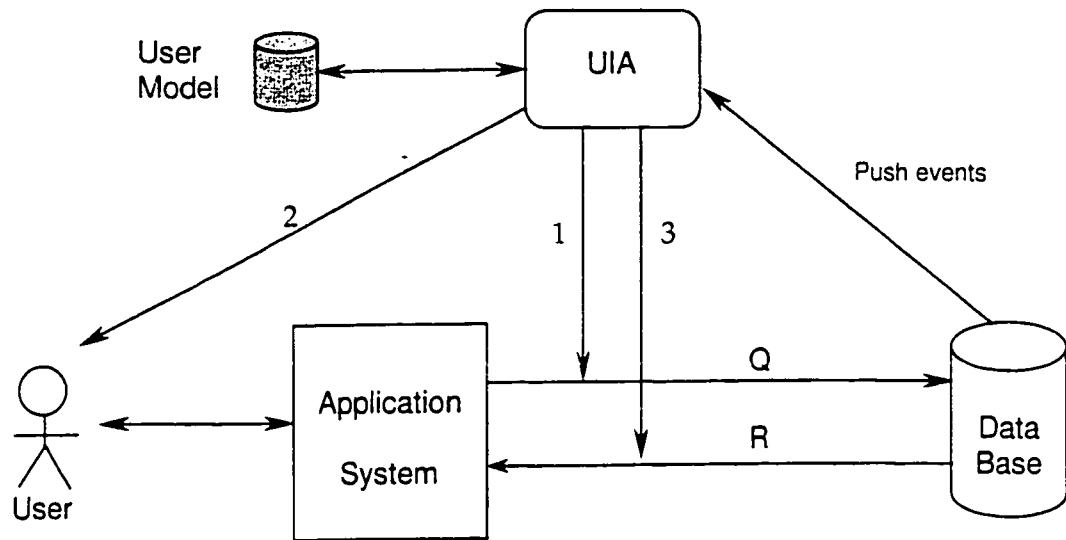
A prototype of UIA is developed as part of this thesis. One reason to consider a prototype is as a “proof of concept”. Through prototyping and testing, we can recognize if there is any problem in the user interface. A prototype enables us to test the concepts before engaging in a full-scale effort to develop an intelligent software system.

A second benefit of prototyping a system is about the development software itself. With so many tools available, the best we can do is to study the features that each tool offers, match them against the requirements demanded by a particular problem, and then use the most appropriate tool. If we find that the tool selected does not contain all the features that the project necessitates, we may either look for another tool (as an alternative or a supplement) or work under the tool’s constraints.

In order to test the tool’s suitability, a prototype can also indicate how well the entire system will perform at run-time. A prototype running slowly suggests that the complete system would be destined to perform unsatisfactorily. Options then would include either trying another tool or redesigning the approach to the problem. In either case, detecting this problem early in the development process is preferable. Finally, by keeping track of the time invested in developing a prototype, we will be better able to estimate the amount of time that the entire project will require.

4.3.2 Functionality of the UIA Prototype

A UIA prototype in this thesis has been proposed for electronic shopping. The prototype maintains a knowledge base about the user model that serves the following three purposes (see Figure 10):



Q denotes Query to the database (SQL in our case)

R denotes Response of the query

Push events are created when the database is updated for proactive response

1: UIA supplements missing information in the query based on the user model

2: UIA provides proactive responsive behavior based on push events

3: UIA can vary the amount and type of information brought to the user

Figure 10: Functionality of the UIA

1. The user model will be used to deal with incompleteness and ambiguities that might arise in the search query specified by the user. In the construction of such a query, the user could afford to leave out certain details for convenience in specification and those details could be filled by the UIA. For example, if the age of a child for whom a gift is to be brought is not specified by the user, based on the context and contents of the user model, the missing parameter will be filled by the UIA to make the query complete. In cases where the agent finds the query ambiguous, it may be able to disambiguate automatically or ask the user for clarification through a dialog.
2. The user model will help the UIA to decide when and what kind of proactive information to present (about sales, new products etc.) to the user. For example, the UIA will alert the user for a birthday gift if his child's birthday (which is stored in the user model) is coming soon (e.g. next month).

3. The user model will help the UIA to filter information and present the search response set to the user in an interactive manner. For example, suppose a catalog search has resulted in a response set with 100 items, the UIA will choose which items to put on the screen.

A user communicates with the UIA through a Graphical User Interface (GUI). The user interface is designed to support interactive communication with the user by employing graphical techniques such as dialog boxes, pop-up windows, and fill-in forms. User can search his interested E-Commerce products through this GUI. The search results from the database will be displayed at the user interface in two possible ways: the user can either browse the results interactively (by clicking the "Next" or "Previous" button in the interface), or use an easy-to-compare table which contains all the results.

In addition to the search capability, the prototype provides a framework at the user interface level for integrating other components such as the software negotiation agent, QoS Management and Security service of the CITR major project (Figure 11). The software sales agent for price negotiation is explained in a companion thesis [Desharnais2 99]. Issues about how to protect users' personal information will be addressed in section 4.5. We will not discuss details about the QoS Management in this thesis. However, users can choose their QoS specifications through the user interface.

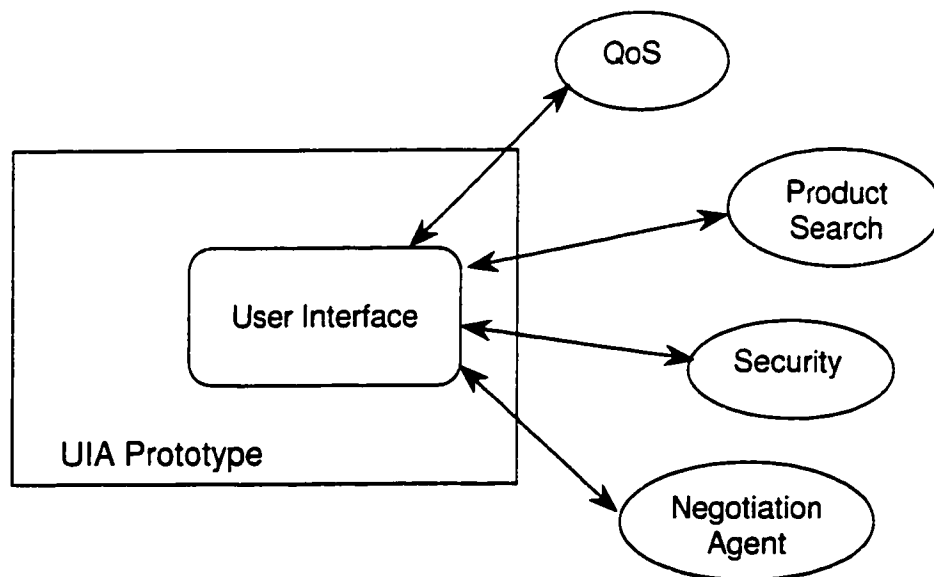


Figure 11: User interface pointing to other components

The UIA contains a shopping cart metaphor (Figure 29) that enables the user to review and modify the contents of his shopping cart during his online shopping process. While browsing, the user may find some interesting items, say items I_1, I_2, \dots, I_n , which he may not decide to buy right then. Therefore he would not like to add to the shopping cart. However, at the end of the shopping, if the total purchasing does not exceed his budget, he may choose from the items I_1, I_2, \dots, I_n . For this purpose, we introduce a “secondary shopping cart” that will hold the items I_1, I_2, \dots, I_n . If the user wants to buy some products from this secondary cart, he can select items from it and add them to his shopping cart. In this manner, the user does not have to go back and search for any of the items I_1, I_2, \dots, I_n again.

When the search is complete and the user is ready to buy, his order will be sent to a remote Sales Agent for validation and that user will receive a confirmation number if the order is accepted. If he wishes to, the user can negotiate the price by sending a quote to the Sales Agent that is lower than the total price [Desharnais2 99]. In this process, the UIA prototype is used to communicate with the vendor’s software Sales Agent for price negotiation interactions.

4.4 User Model for Prototyping the UIA

Building a good user model is essential to help user achieve his goals. User models can be constructed based on two different views of the user’s activities, namely application dependent user activities and application independent activities. For example, the shopping behavior described in section 4.2 can be viewed as application dependent model. In an application independent model, a user can be viewed from many different perspectives [Mayhew 92]. Examples of such perspective are:

1. User’s educational level and experience.
2. User’s psychological characteristics.
3. User’s ability to plan and make selections from a list of choices.
4. The computational facilities available to the user.
5. Common sense knowledge that one expects the others to know.

Our user model does not take all these perspectives into consideration owing to the limited time resources. It is abstracted into the following three categories of specifications:

- I. User's choices or preferences;
- II. A set of constraints that are considered to be relevant by the user;
- III. A set of facts asserted by and "known" to the user.

Such information could be collected by a knowledge engineer or by a learning agent, but this is out of the scope of this thesis work and will not be discussed here. An example from our analysis is shown in Figure 12.

P1.	<i>Everything else being comparable, my choice is a Barbie doll for a baby girl and a toy Match Box car for a baby boy</i>
P2.	<i>My price preference for toys is between \$50-100</i>
P3.	<i>I prefer to buy toys from the store Kid-Mart</i>
P4.	<i>Immediate availability is important</i>
P5.	<i>I am interested in toys manufactured by Fisher-Price for kids under 3</i>
C1.	<i>If order > \$500, the agent should check with me before ordering</i>
C2.	<i>I do not want the toys made in the country M</i>
C3.	<i>No toy guns</i>
C4.	<i>I will not buy from store L</i>
C5.	<i>I will not buy if the price exceeds \$250 for a piece of toy</i>
F1.	<i>Toys in Wal-Mart (store A) are also available in Toyrus (store B)</i>
F2.	<i>I do not like to repeat the same birthday gift (category) given before to my son</i>
F3.	<i>I have more trust in Little Mouse than in Eaton</i>
F4.	<i>Most of my child's toys are produced by Fisher-Price</i>
F5.	<i>I usually go to 3 different stores for price and service comparison before I buy a product which costs more than \$800</i>
P: Preferences; C: Constraints; F: Facts.	

Figure 12: User information to build a user model

Such information is then manually transformed by the designer into a set of *rules* that can be used by a rule engine. In our case, a Java-based rule engine called Jess [Jess 99] is used. Sample rules derived from the analysis of Figure 13 and their application, are shown in Figure 13-16. The UIA can effectively assist the user by making certain inferences and “reasoning” based on a set of rules that capture the user’s needs, preferences and the way he is interacting with the application system.

Rule 1 IF: 1) shopping for toys
 2) brand is not specified in the user’s selection
 3) for girls
 THEN: choose “Barbie doll”

Rule 2 IF: 1) shopping for toys
 2) brand is not specified in the user’s selection
 3) for boys
 THEN: choose “Match Box car”

Rule 3 IF: price range is not specified in the user’s selection
 THEN: choose “between \$50-100”

Rule 4 IF: store name is not specified in the user’s selection
 THEN: choose “Kid-Mart”

Rule 5 IF: the product is not immediately available
 THEN: the user won’t buy it

Rule 6 IF: 1) brand is not specified in the user’s selection
 2) age of the kid is less than 3
 THEN: choose the brand “Fisher-Price”

Figure 13: Rules for user’s choices or preferences in the user model

Rule 7 IF: store ‘A’ is selected
 THEN: products in store ‘A’ are also available in store ‘B’

Rule 8 IF: the user is choosing a birthday gift for his son
 THEN: he usually buys an item not contained in G
 (G: is a set of gifts given to his son before – persistent information)

Rule 9 IF: the product price is greater than \$800
 THEN: present the search results from at least 3 different stores

Figure 14: Rules for facts in the user model

Rule 10 IF: the total purchasing exceeds \$500
THEN: provide a dialog box for confirmation before ordering

Rule 11 IF: the product is made in country M
THEN: the user won't buy it

Rule 12 IF: the product name is "toy gun"
THEN: the user won't buy it

Rule 13 IF: the product is in store L
THEN: the user won't buy it

Rule 14 IF: the product price is greater than \$250
THEN: the user won't buy it

Figure 15: Rules for constraints in the user model

Rules for proactive announcement:

Rule 15 IF: my son's birthday is coming next month
THEN: provide a prompt message for a birthday gift

Rule 16 IF: Rule 7 is fired
THEN: notify the user that the products are available in other stores

Rule 17 IF: the toy price is over 20% of my price preference
THEN: provide a prompt message

Rule 18 IF: the total purchasing exceeds \$1000
THEN: give another 5% discount

Rules for search result presentation:

Rule 19 IF: 1) the user has chosen to display N search results in his profile
2) N is less than M (the size of the total search results)
THEN: N results will be presented to the user

Application of the rules

If the user wants to buy a toy for a baby girl from store A, then different rules are applied in this case:

- Rule 1, 3, 5, 6, 7, 9, 10, 11, 12, 14 are used to refine the user's query
- Rule 16, 17, 18 are used for the proactive notification
- Rule 19 is used for the result presentation

Figure 16: Other rules in the user model

4.5 Privacy of Users' Personal Information

Several surveys suggest that many customers are wary of using electronic shopping because of concerns about their privacy [Privacy 98]. Protection of the *privacy* of the consumer by ensuring proper use of the information contained in his profile (user model) becomes important. At the same time, privacy protection should not be implemented in such a way that it unduly interferes with the free flow of information, one great benefit of the Internet. Ultimately, consumers should feel comfortable with the way their personal information is used and the consumers' ability to control its use.

4.5.1 Protection of the user's personal information

The followings are the essential elements of privacy protection issues [OPA 98]:

- 1) *Adoption and Implementation of a Privacy Policy:* A company engaged in online activities or electronic commerce has the responsibility to adopt and implement a policy for protecting the privacy of individually identifiable information. The company should also take steps that foster the adoption and implementation of effective online privacy policies with other companies they interact, for example, by sharing best practices with business partners.
- 2) *Notice and Disclosure:* A company's privacy policy must be easy to find, read, understand, and available prior to or at the time when consumers' personal information is collected or requested. The policy must state clearly: what information is being collected; the use of that information; possible distribution of that information to third party ; a statement of the company's commitment to data security; and what steps the company takes to control data quality and access. The policy should disclose the consequences, if any, of an individual's refusal to provide information.
- 3) *Choice and Consent:* Consumers should be given the opportunity to make choice with respect to whether and how their personal information is used, either by

businesses with whom they have direct contact or by third parties. Consumers should be provided with simple, readily visible, available, and affordable mechanisms to exercise this option.

- 4) *Data Security:* Companies creating, maintaining, using or disseminating records of identifiable personal information should take appropriate measures to assure its reliability and should take reasonable precautions to protect the data from loss, misuse, alteration or destruction. They should take reasonable steps to assure that third parties to which they transfer such information are aware of these security practices, and that the third parties also take precautions at least to the same extent to protect any transferred information.
- 5) *Data Quality and Access:* Companies should establish appropriate processes or mechanisms so that the personal data they have collected are accurate, complete, and timely. These process and mechanisms should be simple, easy to use, and provide assurance that inaccuracies have been corrected. Other procedures to assure data quality may include use of reliable sources and collection methods, reasonable and appropriate consumer access and correction, and protections against accidental or unauthorized alteration.

4.5.2 Technical Issues

A large number of technologies have been developed to offer solutions to many privacy concerns in the online environment, and they will serve as important tools to protect privacy [Privacy 98].

The Platform for Internet Content Selection (PICS), which was developed by the World Wide Web Consortium to filter out undesirable contents, is being adapted to provide privacy protection. The Platform for Privacy Preferences (P3P) will allow users to set their browsers according to their individual privacy preferences. Once the browser is set, P3P will allow consumers either to avoid web sites that do not provide desired privacy practices or to enter into a negotiation with the web site. P3P will enable web

sites and users to reach an explicit understanding of the privacy practices of those sites and how users' personal data is handled according to their preferences.

Open Profiling Standard (OPS) is another technology to ensure data privacy. OPS is functionally similar to P3P, and it is designed to protect an individual's privacy through the secured storage, secured transport and secured control of user data. In other words, OPS allows the user to control the release of data in a secure manner as opposed to P3P, which provides an agreement between the user and the Web site on how data is compiled.

5. Implementation and Testing

5.1 Implementation Environment

The proposed prototype system is named OSA (Online Shopping Assistant). The OSA has been implemented using the Java programming language, with most development done on a Sun Ultra-1 station running on Solaris 2.5. A number of packages have been used during the development:

- *JDK 1.1.6* (Java Development Kit) for providing a Java development environment
- *Swing 1.1beta3* (GUI toolkit of Java Foundation Classes) for the GUI development
- *JDBC* (Java Database Connectivity) for communication between SQL query and the database
- *Jess 4.3* (Java Expert System Shell) for the rule engine
- *SQL* (Structured Query Language) for the database searches

The system is also connected to the Concordia Oracle database through JDBC. Due to the lack of robust Java integrated development environments (IDEs) for Solaris, an Emacs package for Java has been used for software development and maintenance.

5.2 Software Architecture

The architecture of the system consists of five major components: profile management subsystem, database access utility, result presentation unit, shopping cart metaphor, and Jess inference engine.

The user profile utility is designed to acquire from the user his interests, preferences, and personal information. The database access utility consists of a toy database

backend and a SQL query-generating mechanism intended for test purpose. The presentation unit provides the necessary facilities for displaying and navigating through the results of a query. The shopping cart metaphor allows the user to easily review and modify the items he has selected at any time during the process of his online purchasing. The Jess engine contains a collection of rules and it works closely with other components to adapt the system's behavior based on the user's needs and interactions. The overall architecture of the OSA system is shown in Figure 17.

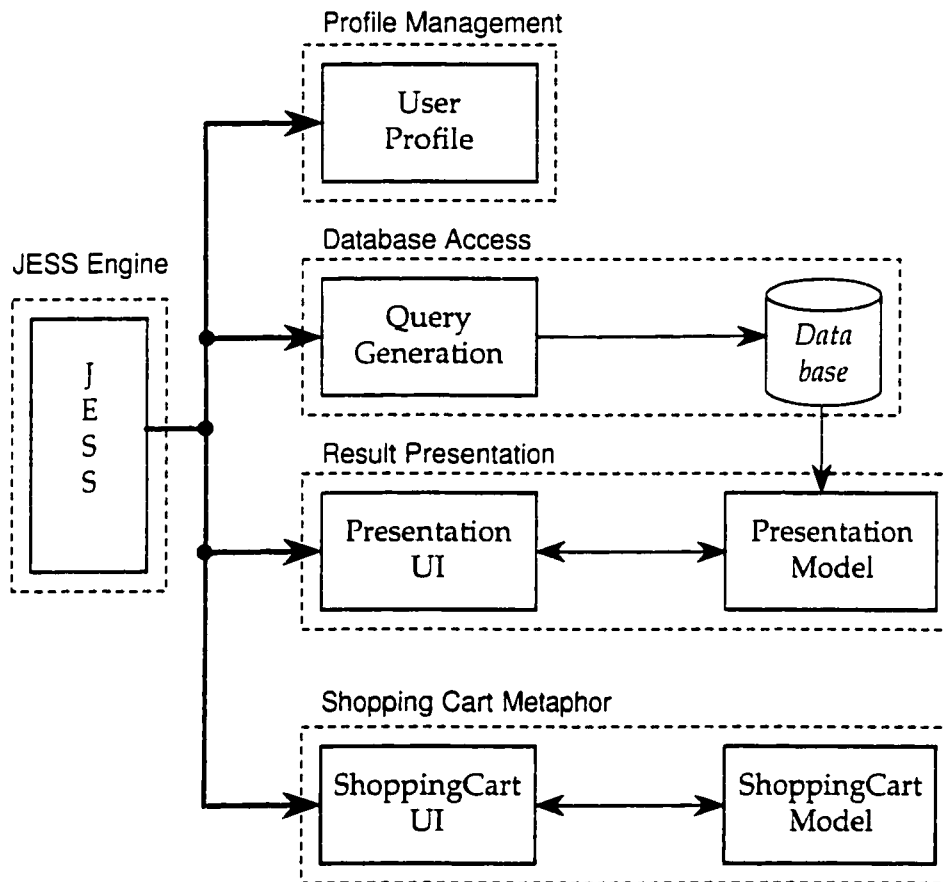


Figure 17: Software architecture

5.3 Subsystem Description

5.3.1 User Profile Management

The user profile is a collection of <attribute, value> pairs and it is accessible and modifiable by the user through the GUI. The availability of the user profile is important for the OSA system. The content of the user profile is composed of the following four parts:

1. *User's personal information* (Figure 18)

Sample personal data could be user's email address, gender, occupation, and the birthday of the people of his choice. The birthday information will be used by the Jess engine to provide proactive prompt message for buying a gift or sending a birthday wish. The operation of the Jess engine that makes use of proactive messages will be described in section 5.3.5.

The screenshot shows a web interface for managing a User Profile (UPP). At the top, there are several tabs: 'UPP (Payment Privacy)', 'Search Queries', 'Matching Toys', 'All Matching Toys', 'Shopping Cart', 'Welcome', 'UPP (Personal Information)', 'UPP (Interests & Preferences)', and 'UPP (Others)'. The 'UPP (Personal Information)' tab is currently selected. Below the tabs, the 'Personal Information' section is displayed. It features a header image of various toys. The form contains the following fields and options:

- Email Address:** A text input field containing 'username@company.ca'.
- Gender:** A dropdown menu with 'male' selected.
- Educational Level:** A dropdown menu with 'Graduate Degree' selected.
- Employment:** A dropdown menu with 'full time' selected.
- Occupation:** A dropdown menu with 'Administrative' selected.
- Birthday and Gender of my kid:** A section with two dropdown menus. The first dropdown is set to 'January' and the second to '01'. To the right, there is a year input field containing '1996'.
- Gender of my kid:** Radio buttons for 'Boy' (selected) and 'Girl'.
- Store User Personal Profile:** A button at the bottom of the form.

Figure 18: Part I of the user profile - personal information

2. User's shopping interests and preferences (Figure 19)

The user's shopping interests are based on his needs and preferences. The user profile contains the brand names of products he prefers along with a rating that shows if the brand is *interesting* or *not interesting* or the user is *indifferent*. The user profile also stores the user's preference for the price over other factors such as immediate availability. A three-point scale is used for this purpose: *very important* / *important* / *unimportant*.

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)		UPP (Interests & Preferences)	
UPP (Others)				

Shopping interests based on product band

Barbie:	<input checked="" type="radio"/> Interested <input type="radio"/> Not interested <input type="radio"/> Indifferent
Fisher-Price:	<input checked="" type="radio"/> Interested <input type="radio"/> Not interested <input type="radio"/> Indifferent
Winnie the Pooh:	<input checked="" type="radio"/> Interested <input type="radio"/> Not interested <input type="radio"/> Indifferent
Hot Wheels:	<input type="radio"/> Interested <input checked="" type="radio"/> Not interested <input type="radio"/> Indifferent
Matchbox:	<input type="radio"/> Interested <input checked="" type="radio"/> Not interested <input type="radio"/> Indifferent
Disney:	<input checked="" type="radio"/> Interested <input type="radio"/> Not interested <input type="radio"/> Indifferent

Preferences and weights for preferences

Price:	<input type="text" value="\$50-100"/>	<input checked="" type="radio"/> very important <input type="radio"/> Important <input type="radio"/> Unimportant
Store:	<input type="text" value="Kid-Mart"/>	<input checked="" type="radio"/> very important <input type="radio"/> Important <input type="radio"/> Unimportant
Availability:	<input type="text" value="Immediately"/>	<input type="radio"/> very important <input checked="" type="radio"/> Important <input type="radio"/> Unimportant






Figure 19: Part II of the user profile - user's interests and preferences

3. *Payment Privacy* (Figure 20, 21)

In order to protect the privacy of a customer, the system provides three levels of access control to the user's personal data such as address, phone number and credit card information: read/write access, read only access, and access denied. Based on the password the user provided, he is allowed different access privileges.

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)	UPP (Others)	

Payment Privacy



First Name:

Last Name:

Address:

Postal Code:

Phone (at home):

Phone (at work):

Credit Card

☐ **Visa**

☒ **Master Card**

☐ **American Express**






Master Card

Expiration Date:

Figure 20: Part III of the user profile - payment privacy (editable mode)

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)		UPP (Others)

Payment Privacy

First Name:

Last Name:

Address:

Postal Code:

Phone (at home):

Phone (at work):

Credit Card:

☐ Visa

☒ Master Card

☐ American Express

Master Card

Expiration Date:

Figure 21: Part III of the user profile - payment privacy (read only mode)

4. Others (Figure 22)

There are several other inputs that can be obtained from users to create user profiles:

- *Methods of information filtering* and *Display Mode* useful to present the search results
- *Triggers* and *triggered proactive messages* which alert the user if the product's price is over certain level or notify him the release of new products or special offers based on his shopping interests
- *QoS Requirement* intended for the QoS management subproject

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)	UPP (Others)	

Methods for information filtering

☐ show all the matching toys
☒ show the first items which satisfy my shopping preferences

Display Mode

☐ Display the detailed search results one by one
☒ Display ALL the search results in a table

Notification Messages

☒ Notify me if other stores have the same product.
☒ Notify me about the special offers based on my interests.

QoS Requirement

☒ Best try effect
☐ User specified parameters:

Access time: seconds
 Quality of VRML world: ☐ Normal ☐ Good ☐ Excellent

Figure 22: Part IV of the user profile – others

5.3.2 Database Access and SQL Generation Utilities

The OSA system provides direct manipulation on GUI to enable a user to search for his interested toys in the database. It then generates equivalent queries in SQL and uses JDBC to communicate to the database. If the user's input queries are incomplete, the system will attempt to fill the missing attributes in the queries based on the contents of that user's profile.

5.3.2.1 Indexing of Toy Items

In order to generate a test database, the following relational schema is used:

- *Id* identification number of the toy
- *Name* name of the toy
- *Price* price of the toy
- *Brand* brand which the toy belongs to
- *Category* category which the toy belongs to
- *Gender* gender for which the toy is designed
- *Age1* the lower bound of the age range
- *Age2* the upper bound of the age range

For example, a toy “Barbie doll” might have the following record structure:

<i>Id</i>	14541
<i>Name</i>	Romantic Rose Bride Barbie
<i>Price</i>	179.00
<i>Brand</i>	Barbie
<i>Category</i>	Collectibles Toys
<i>Gender</i>	Girl
<i>Age1</i>	2
<i>Age2</i>	8

5.3.2.2 Query Generation From The User Selection

A user can choose from different brands, categories, gender, age and price range using the GUI as shown in Figure 23. The system converts the GUI based selection to SQL queries and sends the queries to the database using “*prepared statements*”, a feature provided by JDBC [CoreJava 98]. Prepared statement is a “precompiled” SQL query type that may contain parameters. Each parameter is indicated with a question mark (?). A value in the form of text string can be used to substitute the parameter. By substituting the parameter different queries of the same prepared statement type can be

easily generated. It is more efficient to use a prepared statement for sending SQL query to the database because the query can be executed without having to compile it first.

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)		UPP (Others)

Search Queries

1. Select as many or as few brands, categories as you wish

Brand:

- Any
- Barbie**
- Fisher-Price
- Winnie the Pooh
- Hot Wheels
- Matchbox
- Disney

Category:

- Any
- Collectibles Toys
- Dolls
- Games
- Playsets
- Garden & Outdoor

2. Select the gender, age and price range

☒ for Girl

☐ for Boy

☐ for Both

Age: **2 - 3 yrs**

Price Range (in \$): **All prices**

3. Click the button on the right to:

Show Matching Toys

Figure 23: GUI for search query generation

Considering the query for all toys with the brand "Fisher-Price", the normal SQL query is:

```
SELECT id, name, price
FROM Toys
WHERE brand = "Fisher-Price"
```

The prepared query for the above SQL becomes:

```
String brandQuery =
    "SELECT id, name, price " +
    "FROM Toys " +
```

```

"WHERE brand = ?";
PreparedStatement brandQueryStmt
= con.prepareStatement(brandQuery);

```

where the connection object con provokes the JDBC drivers to manage SQL queries.

Before executing the prepared statement, we must bind the parameters to actual values with a set command. There are different set statements for the various data types. In the above case we want to set a string:

```
BrandQueryStmt.setString(1, brand);
```

where 1 denotes the first ? in the prepared query, brand is the value that we want to assign to the parameter.

There are sixteen different prepared queries in the OSA system, one for each of the sixteen cases shown in Table 3 below:

Query Type	Brand	Category	Gender	Price	Age
Query 1	Specified	Specified	Specified	Specified	Specified
Query 2	Specified	Specified	Specified	Any	Specified
Query 3	Specified	Specified	Any	Specified	Specified
Query 4	Specified	Specified	Any	Any	Specified
Query 5	Specified	Any	Specified	Specified	Specified
Query 6	Specified	Any	Specified	Any	Specified
Query 7	Specified	Any	Any	Specified	Specified
Query 8	Specified	Any	Any	Any	Specified
Query 9	Any	Specified	Specified	Specified	Specified
Query 10	Any	Specified	Specified	Any	Specified
Query 11	Any	Specified	Any	Specified	Specified
Query 12	Any	Specified	Any	Any	Specified
Query 13	Any	Any	Specified	Specified	Specified
Query 14	Any	Any	Specified	Any	Specified
Query 15	Any	Any	Any	Specified	Specified
Query 16	Any	Any	Any	Any	Specified

Table 3: Sixteen different types of queries

5.3.2.3 Query Generation From The User Profile

As an intelligent user interface agent, the OSA system not only responds passively to the user's commands and queries, it is also capable of taking its own initiative in order to volunteer information to fill up incomplete or unclear user queries. For example, if the user has not explicitly chosen the brands, categories, gender, age and price range on the GUI, the system is able to extract the corresponding parts from the user profile and replace the missing ones appropriately to make the queries complete. Below, we describe them in detail for each field:

1. If the user has not selected the *brand* field, the system uses those brands that have been marked as *interested* to the user in the user profile. In case the user is not interested in any specific brand, the system will use those brands that have been marked as *indifferent* to the user.
2. If the user has not selected the *category* field, the system assumes the default value of *all* categories.
3. If the user has not selected the *gender* field, the gender of his child in the user profile is used.
4. If the user has not selected the *age* field, the age of his child is calculated based on the kid's birthday in the user profile.
5. If the user has not selected the *price range* field, the system uses his preferred price range in the user profile.

5.3.3 Result Presentation

The toys retrieved from the database by matching the SQL query can be viewed interactively. If the button "show all matching toys" in the user profile has been selected, all toys are presented. Otherwise the specified number of toys which satisfy his shopping preferences are displayed. The OSA system provides two different ways of presenting the search results: one is to display all the matching toys in a table which enables the user to compare them easily (Figure 25); the other supplies detailed

information such as the name, price, item number, an image and a description text of the toy that matches the user's query (Figure 24), and allows the user to browse them one after another by clicking the "Next" or "Previous" button on the GUI. If there is no toy matching the user's search criterion, the system will generate an alert message, and all the buttons ("Previous", "Next", "Add to shopping cart") will be disabled.

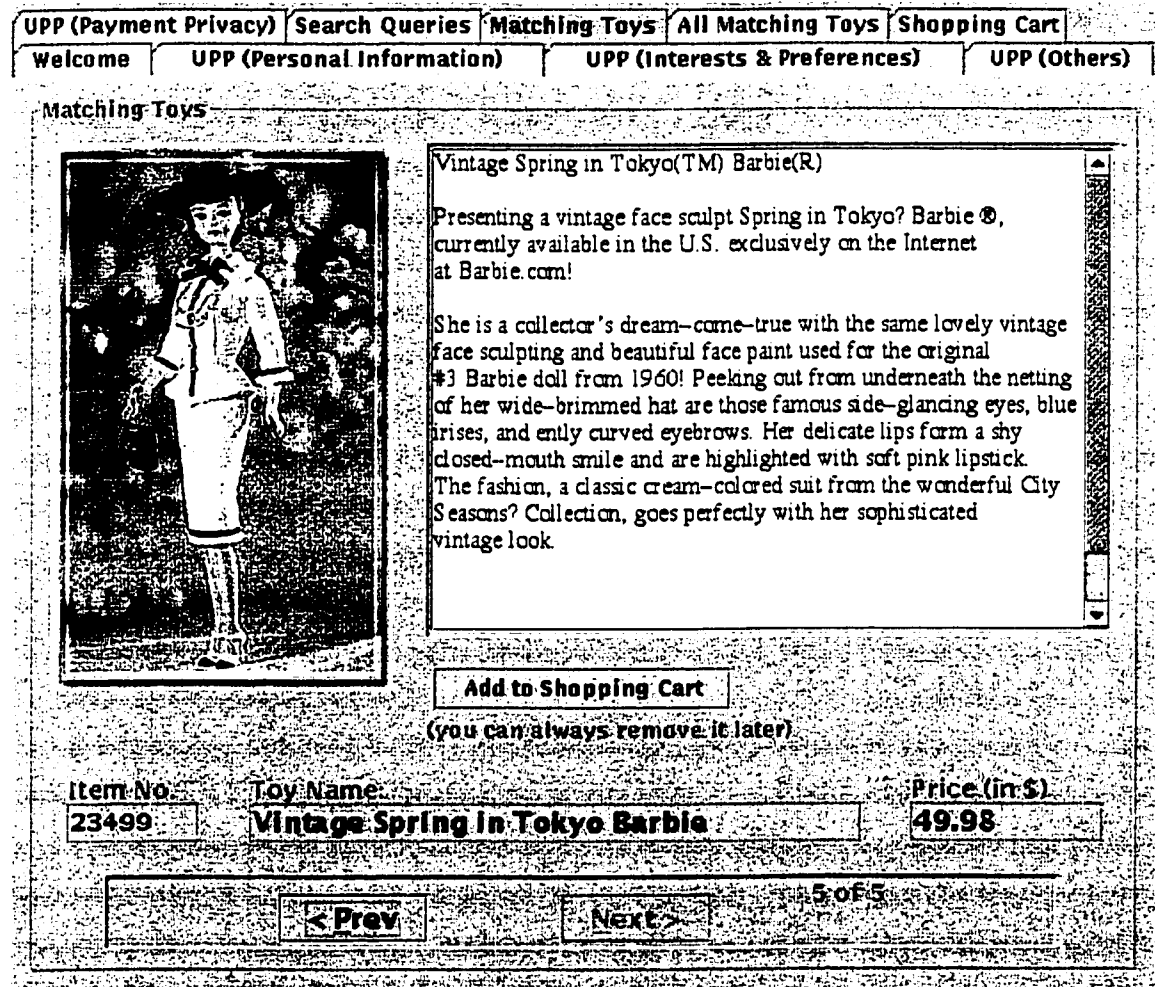



Figure 24: Presentation window with detailed product information

The underlying architecture of the presentation module is a simplified variant of MVC [MVC 99], known as the *model-delegate*. In MVC design pattern, there are three communicating objects as shown in Figure 26: the *model*, *view* and *controller*. The model is essentially a data model. The view is the visual representation of that data. The controller specifies how to handle user input. When a model changes, it notifies all

views that depend on it. The aim of MVC is to provide a high-level object-oriented structure that can be used as the basis for interactive applications.

UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)	UPP (Others)	

All Matching Toys		
Here are all the toys matching your needs		
		
Item No	Toy Name	Unit Price (in \$)
21414	Heartstring Angel Barbie	89.00
15683	Summer Splendor Barbie	79.00
15204	Autumn Glory Barbie	79.00
14541	Romantic Rose Bride Barbie	179.00
23499	Vintage Spring in Tokyo Barbie	49.98

Click the button on the right to: [show detailed information](#)

Figure 25: Presentation table containing all the search results

The model-delegate diagram combines the view and the controller into a single element: *UI delegate*, which is responsible for drawing components to the screen and handling input events. Bundling graphics capabilities and event handling together is somewhat easy in Java, since much of the event handling is taken care of in Java AWT (Abstract Window Toolkit) [AWT 97]. This way, the communication in MVC becomes a two-way interaction as shown in Figure 27. In the OSA system, whenever the presentation model (the model) changes as a result of a user query, the presentation UI (the UI delegate) is automatically updated to reflect the latest changes.

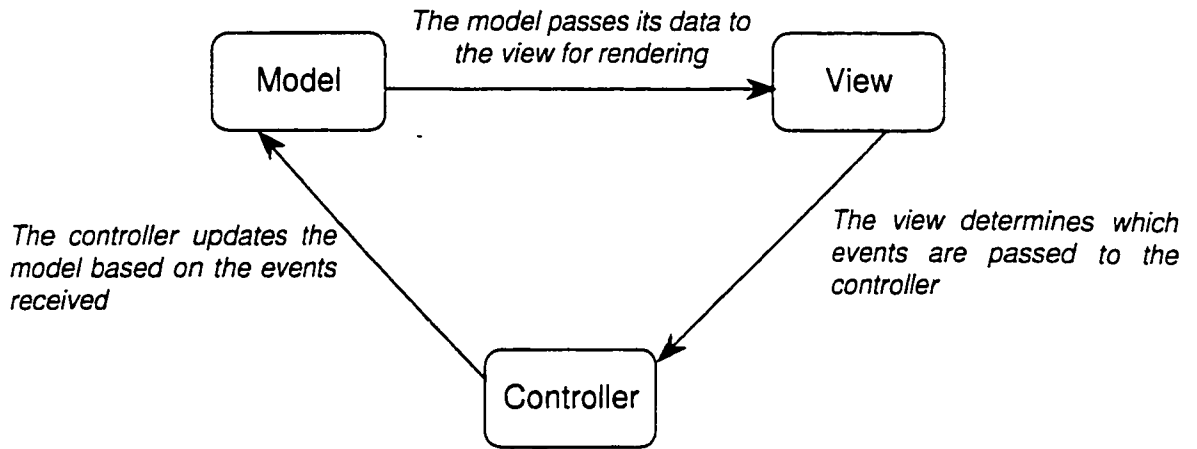


Figure 26: Communication through the Model-View-Controller architecture

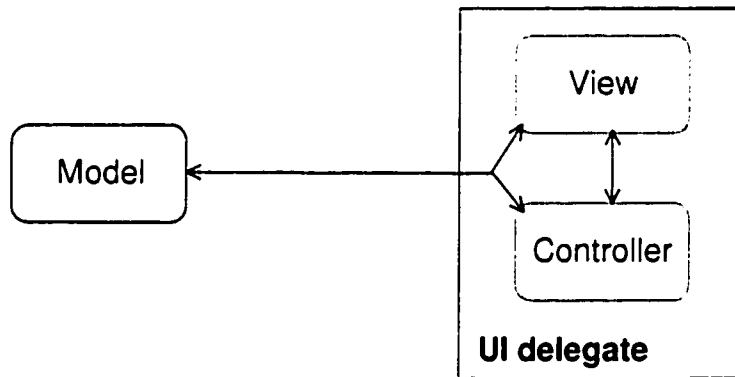


Figure 27: The Model-delegate combines View and Controller into UI-delegate


5.3.4 Shopping Cart Metaphor

The shopping cart metaphor is the most common checkout method used for online shopping [Lohse 98] and we have used it in the OSA system (Figure 28). It allows the user to review the contents of the shopping cart (items selected by users for buying) at any time during the shopping process.


If the user is satisfied with the toy displayed in the result window, it is always easy to put that item to his shopping cart by clicking the “Add to shopping cart” button in the result window. On the other hand, it is also very easy to remove some items from the shopping cart. The user simply needs to modify the quantity field of the shopping cart table. If the quantity is set to zero, then the corresponding item is automatically deleted from the shopping cart.


UPP (Payment Privacy)	Search Queries	Matching Toys	All Matching Toys	Shopping Cart
Welcome	UPP (Personal Information)	UPP (Interests & Preferences)	UPP (Others)	

Shopping Cart

Here are the items in your  **Shopping Cart**

Item No	Toy Name	Qty	Unit Price	Amount
15204	Autumn Glory Barbie	1	79.00	79.00
23499	Vintage Spring in Tokyo Barbie	1	49.98	49.98
15683	Summer Splendor Barbie	1	79.00	79.00

 To REMOVE an item from your cart, enter '0' in the 'Qty' box.

 To CHANGE the quantity of an item, enter the new quantity.

Total (before discount)	\$207.98	You can enter the amount that you wish to pay in My Offer field. However, by doing so, your order might be REJECTED! Please see Negotiation Rules.
GST	\$15.60	
QST	\$14.56	
Grand Total	\$238.14	
My Offer	\$200.00	

Send Order / Negotiate

Figure 28: Shopping cart window

The shopping cart metaphor is implemented using the model-delegate architecture described in the previous section. When the shoppingCart model (the model) changes as the user adds or removes items to or from the shopping cart, the shoppingCart UI (the UI delegate) is automatically updated to reflect the latest modifications on GUI.

5.3.5 Jess Engine and Rules

Jess is an expert system shell written entirely in Java [Jess 99]. It was originally inspired by CLIPS, but has grown into a complete, distinct Java-influenced environment to allow easy integration with Java code. Jess can be used in two overlapping ways. First, it is a rule engine which supports the development of rule-based expert system. Second, it is also a general-purpose scripting language which can directly access all Java classes and libraries. While Java code generally must be compiled before it can be run, a line of Jess code can be executed interpretively. This allows us to experiment with Java APIs interactively, and build large programs incrementally.

Jess is useful in a wide range of situations. Our application here is to develop intelligent agents that have the capacity to “reason” effectively using knowledge that is stored in the form of declarative rules. Rules in Jess are somewhat like the IF...THEN... statement of other programming languages. In operation, Jess constantly tests and executes the specified action if the predicate in the IF part becomes true.

There are several ways to integrate Jess and Java code: We can embed Jess in the Java applications, or write Java classes and add them to Jess so that they will appear as a part of the Jess language, or manipulate Java objects directly from Jess. To have more control over Jess from our Java program, we use `jess.Rete` class's `executeCommand(String cmd)` method as shown below (The `Rete` class contains the expert system engine):

```
try
{
    ...
    rete.executeCommand("(clear)");
    rete.executeCommand("(batch jess/toy/toyrule.clp)");
    rete.executeCommand("(reset)");
    rete.executeCommand("(assert (psd 3))");
    rete.executeCommand("(run)");
}
catch (ReteException e)
{
    e.printStackTrace(nd.stderr());
}
```

In the OSA system, the Jess engine interacts with all the other subsystems to manifest a level of intelligence of the user interface agent. The followings are the different uses for Jess in our prototype:

1. The Jess engine can proactively alert the user about a forthcoming event (e.g. birthday) based on the user profile.
2. The Jess engine can be used to authenticate a user beyond the password entry when necessary. The extra information requested could be verified with the data stored in the user profile.
3. The Jess engine can be used to choose boundaries for terms like “slightly” higher (e.g. 0-5% over) or “higher” (e.g. 5-20% over) or “too high” (e.g. 20%+ over) based on the user’s preferences. If the price is only a bit higher, this provides the possibility to negotiate the product price with the vendor. Then a negotiating agent can be activated [Desharnais2 99].
4. The Jess engine can change the number of products to be displayed in the resulting window (one, two, three, etc.) based on the user’s needs and other related information.
5. On behalf of the vendor, the Jess engine is able to determine whether to give the user more discounts based on the total amount of the purchase, or otherwise.

5.4 Testing of Querying and UIA Assistance

Testing is done in two phases. The first phase is designed to test the system's function in converting the user's input to SQL query and communicating to the database server through JDBC. In this phase, all the sixteen different query types mentioned in section 5.3.2.2 were tested under the assumption that the user had explicitly chosen all search fields (brands, categories, gender, age and price range) on the GUI. There was no assistance of the UIA at this stage. The test results showed that the search response set for each query type did match the user's selections. This phase of testing served as the starting point for further testing of UIA's capability to supplement missing information in the search query based on the user profile, which was done in the second phase. In this phase, the user is assumed to have not explicitly selected *at least* one search field in the input. The various query types were tested according to the following task sets:

Task set 1:

- 1.1 search the products without specifying interests, preferences, and/or constraints in the user profile

Task set 2:

- 2.1 specify interests, preferences and/or constraints in the user profile
- 2.2 do the search

Task set 3:

- 3.1 specify interests, preferences and/or constraints in the user profile
- 3.2 do the search
- 3.3 change the contents of the user profile
- 3.4 search the products database again

The test results for each task set are summarized in Table 4, from which we can see that the UIA is able to supply missing information at the input and this improves the retrieval performance.

	Methods employed to supply Missing information at the input	Test results
task set 1	<p>The missing fields in the query were filled with default values in the user profile:</p> <ul style="list-style-type: none"> -- all <u>brands</u> -- all <u>categories</u> -- all <u>prices</u> -- for both <u>genders</u> 	<p>If the user had not chosen one particular age range, a message box would be popped up to alert the user to do so. This conforms to the constraint that the "Age" field must be specified for all sixteen query types (Table 3).</p> <p>The search response set included all products within one age range from the sample database.</p>
task set 2	<p>The missing fields in the query were filled according to the contents of the user profile:</p> <ul style="list-style-type: none"> -- those <u>brands</u> marked as "interested" to the user -- all <u>categories</u> -- the user's preferred <u>price</u> range -- the <u>gender</u> of the user's child -- the <u>age</u> of the user's child 	<p>If the user had marked all the brands as "not interested", a message box would be displayed to ask the user to choose at least one brand as "interested" or "indifferent".</p> <p>The search results matched the user's interests and personal situation.</p>
task set 3	<p>The missing fields in the query were supplemented according to the user's latest changes in his user profile in the course of the shopping.</p>	<p>The user could change his user profile during the shopping process. UIA was able to keep track of the updated user profile. The search response sets reflected the changes in the user profile.</p>

Table 4: Test results for different task sets

We also test the recall and precision using the sixteen types of search queries. Recall is the proportion of all relevant documents that are actually retrieved as a result of a query. Precision is the proportion of a retrieved set of documents that is actually relevant [Belkin 92]. Because the size of our sample database is small, both the recall and the precision is 100% in our testing.

5.5 Testing of Shopping Cart Metaphor

Testing the shopping cart metaphor was conducted for the three features described below:

Feature 1:

The user clicks a button and adds the item to the shopping cart. It is possible that the user clicks the button twice (or more many times) by mistake or to make sure the item is added to the shopping cart. The implication here is that the shopping cart contains only “one” item for each product type.

Feature 2:

The user really wants more items of the same type, then he can edit the shopping cart directly.

Feature 3:

The contents of the shopping cart should be cleared when the user sends his order or logs off the current shopping session.

For feature 1, if the user’s selected item has not been added to the shopping cart before, it will be put there when the user clicks the “Add to shopping cart” button in his online shopping process (Figure 24). Otherwise, it will not be added to the shopping cart when the user clicks that button. The OSA system is able to detect whether the user’s selected items have already been in the shopping cart or not, thus release the user from memorizing the items he has already selected or looking back and forth to check with the contents of the shopping cart. The test results showed that the system acted as expected.

The default quantity for all the items in the shopping cart is “one”. If the user needs to change the product quantity or remove an item from the shopping cart, he has the options to do so. For feature 2, the user can modify the shopping cart by typing his desired quantity in the “qty” field (Figure 28) for that item. If he types a zero, that item is automatically removed from the shopping cart. The test results showed that the user could easily modify the contents of the shopping cart. When the “qty” field was updated by the user, all other fields like “Amount”, “Total”, “GST”, “QST”, and “Grand Total” would also changed accordingly and correctly.

Testing for feature 3 was performed under the following three scenarios (Table 5):

1. The user clicks the “Send Order” button (Figure 28) to purchase the items in the shopping cart, and then continues to search for more product information and may place other orders in the current session.
2. The user clicks the “send Order” button to purchase the items in the shopping cart, finishes the current session, then logs on to the OSA system again to start another search session.
3. The user finishes the current search session without clicking the “Send Order” button, which means that he will not buy those items in the shopping cart. He then logs on to the OSA system again to start another search session.

	Send order	Log off current session	Log on next session
Scenario 1	Yes	No	—
Scenario 2	Yes	Yes	Yes
Scenario 3	No	Yes	Yes

Table 5: Scenarios for testing the shopping cart

The test results for feature 3 were satisfactory with all the fields in the shopping cart cleared properly in different situations mentioned above.

6. Summary and Conclusions

Electronic commerce opens up new opportunities and challenges in retail markets. More than ever before, consumers are able to compare an incredible amount of value added products without the need to leave their homes. However, current online businesses are faced with increasing competition for people's time and attention. To be successful, electronic shopping systems need to provide "exceptional" value and service based on customer's needs through personalized service to their customers. We believe that intelligent agents can help to create that kind of personalized service to help consumers effectively navigate the abundance of product information. These services may come in different forms such as delivering information to the customers according to their interests, meeting the time or resource constraints and expectations; personalized product recommendations; sales promotions tailored to individuals; and bargaining the price of a purchase. In this thesis, we have explored the use of an intelligent user interface agent that addresses one issue, namely providing personalized service to the user by knowing the user's preferences.

6.1 Conclusions

The Consumer Buying Behavior (CBB) model from MIT media lab [Guttman, 98] comprises the actions and decisions involved in various stages of buying and using goods and services from the commerce perspective. The CBB model consists of six different stages which are (1) Need identification (2) Product brokering (3) Merchant brokering (4) Negotiation (5) Purchase and delivery (6) Service and evaluation. These six stages represent an approximation and simplification of the complex buying behaviors. The thesis work is based on this CBB model. In order to provide personalized services in various stages of the CBB model, we have studied the customer's buying behaviors. Users in their single shopping experience are categorized

into one of the three task-oriented groups: (I) Comparative Shopping (II) Planned Shopping (III) Browsing Based Shopping. Each group is relevant to different stages of the CBB model, and has different requirements and constraints.

To effectively assist the user, a user interface agent (UIA) should have a model of the user, which is an explicit representation of relevant properties of that user with regard to a priori known tasks. User models can be constructed based on application dependent and application independent attributes. Our user model focuses on three aspects: (1) User's choices or preferences (2) constraints (3) facts asserted by and known to the user. This user model is used for the purpose of:

- I. dealing with incompleteness and ambiguities that might arise in the search query and using context based substitution;
- II. helping the UIA to decide when and what kind of proactive information to present to the user;
- III. enabling the UIA to filter information and present the search response set to the user in an interactive way.

The UIA has been implemented as an operational prototype on a Java platform for electronic shopping. The UIA maintains a knowledge base in the form of a user's personal profile. The user profile contains information about user's shopping characteristics, choices, preferences, constraints and personal data. Such a profile is initially specified by the user and the user can also update his profile at any time during his interaction with the system. Users can search their interested products through a conventional GUI. The UIA then generates equivalent SQL queries and communicates with a remote product database. While generating SQL, the UIA is able to supplement missing information in the query or refine the user request according to the user profile. The UIA is designed to proactively assist the user, by making certain inferences based on the user's needs and interactions with the system. The inferencing needed for this purpose is achieved by using a set of rules that are invoked by JESS, a Java based rule engine. The UIA contains a shopping cart metaphor that enables the user to easily review and modify the items he has chosen during his online shopping process. The UIA prototype provides an interface connecting with the vendor's software sales agent

for price negotiations. A set of test cases were used for testing the sixteen types of search queries, and the UIA was formed to properly resolve the incompleteness in the queries, based on the contents of the user profile. The three purposes (I - III) mentioned above have been demonstrated by the prototype. This is expected to improve the effectiveness of a user interface for E-Commerce applications.

6.2 Future Work

The UIA prototype requires extensive usability testing in order to determine its usefulness in assisting users for e-commerce applications. Is the system useful, complete and friendly enough? Is the system efficient in retrieving and displaying information? These questions need to be answered in the usability testing.

The secondary cart (mentioned in section 4.3.2) relieves the user from repeatedly searching the products if he wants to buy items that have not been added to his shopping cart. It can be implemented as part of the shopping cart metaphor of the UIA prototype with minor efforts, if necessary.

Acquisition of the user model is an important issue to assist users in performing their tasks of selection, ordering and receiving of products or services in an electronic commerce environment. There are different ways to elicit the user model. In this thesis, we only introduce one approach — the use of a GUI-based user interface by a domain specialist to collect the user profile. However, to be more user-friendly and interactive, the needed information can be elicited through “dialogs” between the user and the UIA in an incremental fashion. The dialogs and conversations combined with natural language understanding, speech recognition and speech synthesis technologies will bring naturalness, flexibility and ultimately effective support to online shopping.

A learning agent can also help to build the user model implicitly by observing users’ behaviors. It will be beneficial to study under which situations learning can take place, and which learning algorithms are appropriate for e-commerce applications. The GUI

based user interface presented in this thesis can serve as a starting point to determine what items can be automatically learned by a software agent.

We have classified three different shopping types as well as different requirements for each shopping type in this thesis. The problem of how to create a user model for each shopping type according to the requirements and adapt the system's behavior based on the user model remains.

7. References

1. [Wooldridge 95] M. Wooldridge and N.R. Jennings, "Agent Theories, Architectures, and Languages: a Survey", *Intelligent Agents*, Wooldridge and Jennings Eds. , Berlin: Springer-Verlag
2. [Mandel 97] T. Mandel, "Social User Interface and Intelligent Agents", *The Elements of User Interface Design*, Wiley Computer Publishing, N.Y. 1997, pages 357-381
3. [Sullivan 91] J. W. Sullivan and S. W. Tyler, "Intelligent User Interfaces". *ACM Press* N.Y. 1991
4. [Georganas 98] N.D. Georganas, "Distributed Virtual Environment User Interface and Intelligent Agents", University of Ottawa, 1998
URL: < http://www.mcrlab.uottawa.ca/research/CITR_98.htm>
5. [Guttman 98] R. H. Guttman, A. G. Moukas, and P. Maes. "Agents-mediated Electronic Commerce: A Survey". *Knowledge Engineering Review*, June 1998
6. [Maes 98] P. Maes, R. H. Guttman, A.G. Moukas, "Agents That Buy And Sell" *Communications of the ACM* 42(3), 1998, pages 81-91
7. [Maes 94] P. Maes. "Agents that reduce work and information overload" *Communications of the ACM* 37(7), 1994, pages 30-40
8. [URL1 99] PersonaLogic URL: <http://www.personalogic.com/>
9. [URL2 99] Firefly URL: <http://www.firefly.com/>

10. [URL3 99] BargainFinder URL : <http://bf.cstar.ac.com/bf>
11. [URL4 99] Jango URL: <http://www.jango.com/>
12. [URL5 99] Tete-a-Tete URL: <http://ecommerce.media.mit.edu/Tete-a-Tete/>
13. [Morris 87] A. Morris, "Expert Systems - Interface Insight", *People and Computers III: Proceedings of the Third Conference of the British Computer Society, Human Computer Interaction Specialist Group*, 1987, pages 307-324
14. [Browne 90] D. Browne, M. Norman, E. Adhami, "Methods for Building Adaptive Systems", *Adaptive User Interfaces*, Academic Press, London, 1990, pages 85-130
15. [Dieterich 93] H. Dieterich, U. Malinowski, T. Kuhme, M. Schneider-Hufschmidt, "State of the Art in Adaptive User Interfaces", *Adaptive User Interfaces: Principles and Practice*, Elsevier Science Publishers, 1993, pages 13-48
16. [Lashkari 94] Lashkari, Y., Metral, M., and Maes, P. "Collaborative Interface Agents", *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994
17. [Brajnijk 87] G. Brajnijk, G. Guida, C. Tasso, "User Modeling In Intelligent Information Retrieval", *Information Processing & Management*, 23(4), pages 305-320
18. [Strachan 97] L. Strachan, J. Anderson, M. Sneesby, M. Evans, "Pragmatic User Modeling in a Commercial Software System", *User Modeling: Proceedings of the Sixth International Conference*, 1997
19. [CoreJava 98] C. S. Horstmann, G. Cornell, Chapter 4: Database Connectivity: JDBC in *Core Java 1.1 Volume II — Advanced Features*, Sun Microsystems Press, 1998, pages 187-223

20. [MVC 99] Model View Controller, available online at <http://bmrc.berkeley.edu/courseware/cs160/fall97/lectures/11-26-97>
21. [AWT 97] J. Zukowski, Chapter 1: Abstract Window Toolkit Overview in *Java AWT Reference*, O'Reilly, 1997
22. [Jess 99] Ernest J. Friedman-Hill, Sandia National Laboratories, Homepage of Jess - the Java Expert System Shell, <http://herzberg.ca.sandia.gov/jess/>
23. [Mayhew 92] Deborah J. Mayhew. "Principles and Guidelines in Software User Interface Design". Prentice Hall Inc., New Jersey, 1992
24. [Desharnais1 99] P. Desharnais, J. Lu and T. Radhakrishnan, "Exploring Agent Support at the User Interface in E-Commerce Applications", *International Symposium on Electronic Commerce*, Beijing, May 1999
25. [Desharnais2 99] P. Desharnais, in Preparation of Master's Thesis about "Software Sales Agent for Price Negotiations", Department of Computer Science, Concordia University, 1999
26. [Belkin 92] N. J. Belkin and W. B. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, 35(2), 1992, pages 29-38
27. [Privacy 98] "Privacy and Electronic Commerce", Draft prepared by United States, June 1998, available online at <http://www.doc.gov/ecommerce/privacy.htm>
28. [OPA 98] "Privacy Guidelines Commentary" Submitted with the Comments of the Online Privacy Alliance On the Draft International Safe Harbor Principles by Online Privacy Alliance, November 1998, available online at <http://www.privacyalliance.org/news/12031998-4.shtml>

29. [Lohse 98] G. L. Lohse, P. Spiller, "Electronic Shopping", *Communications of the ACM* 41(7), 1998, pages 81-87