Université de Montréal

XMLFinder : an Intelligent Agent Based on CBR for E-Commerce

Par:

YanPing Ma

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître en Informatique

Juillet, 2001

QA
76
N54
2001
v.030

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé:

XMLFinder : an Intelligent Agent Based on CBR for E-Commerce

Présenté par:

YanPing Ma

a été évalué par un jury composé des personnes suivantes:

Claude Frasson	Président-rapporteur
Esma Aïmeur	Directrice de recherche
Sahraoui Houari	Membre

Mémoire accepté le 4 Septembre 2001

Acknowledgement

This work has been a challenging experience through which I learned about frontiers of research and many exiting technologies. Through the year, I received a lot of support from the professors and friends in our department.

First, I would like to thank Prof. Esma Aïmeur, my supervisor, whose continued guidance, encouragement, enthusiasm, and support are greatly appreciated. I also owe my gratitude to Prof. Gilles Brassard for his great comments on my project and published paper. Great thanks are also given to Prof. Jian-Yun Nie, Prof. Jan Gecsei and Dr. Jalal Kia for their encouragement and help.

Many thanks to all of my friends for their encouragement and thoughtful discussions. I also would like to thank the people who evaluated my system, and gave me their valuable comments.

Great thanks to my husband for his support, patience and faith in me, and to my lovely son for his understanding and the joy he brings to my life.

Résumé

Le commerce électronique est un aspect de l'Internet qui croît très rapidement. Il existe divers types de sites d'achats en ligne qui aident des consommateurs à trouver les produits recherchés. L'agent intelligent est parmi les technologies les plus prometteuses dans les systèmes de recommandation, et le raisonnement à base de cas (CBR, Case-Based Reasoning) est une technique hautement efficace pour développer de tels agents. Nous proposons dans ce mémoire un agent intelligent basé sur le CBR, appelé XMLFinder qui recommande des produits aux clients en ligne.

Il y a deux caractéristiques principales dans ce système. La première, en utilisant la technologie de CBR, concerne la manière dont le système permet aux consommateurs d'évaluer et d'adapter leurs exigences aux produits. Habituellement, obtenir des résultats similaires est plus important que l'obtention de résultats totalement correspondants, parce que la plupart des clients en ligne ne sont pas capable de formuler une requête exacte décrivant leurs exigences. Une nouvelle notion appelée l'insuffisance ("deficiency"), est présentée. L'insuffisance d'un attribut mesure à quel point le système considère cet attribut approprié aux préférences de la part de l'utilisateur. L'heuristique proposée est exprimée comme suit : si le client ne choisit pas le plus "déficient" des cas considérés, alors l'importance des aspects le plus déficient doit diminuer et l'importance l'aspect préféré doit augmenter. La justification intuitive derrière cette heuristique est que les clients ont tendance à choisir d'abord l'aspect auquel ils attachent le plus d'importance. C'est ce qui constitue le profil à court terme dans ce mémoire. Cela permet à notre système, en combinant cette approche avec le profil à long terme (telles que les contraintes des consommateurs), de fournir des recommandations de manière plus efficace.

La deuxième caractéristique est l'utilisation de XML (eXtensible Markup Language). XML est un langage de transmission permettant à de grands paquets d'information formatée d'être échangés réduisant de ce fait le trafic de réseau. XMLFinder est construit sur une architecture de trois couches, dans laquelle les documents XML jouent un rôle dans chaque couche. Étant une alternative possible de HTML, XML aide le Web a supporter les applications intelligentes. XSL (eXtensible Stylesheet Language) est employé pour fournir une séparation nette entre le contenu lui-même (XML) et le format spécifique exigé par un programme ou un document cible. Ainsi un emballage (wrapper) est fourni pour le catalogue en un fichier XML traduit par XSL. La représentation des cas et des similitudes en XML est une nouvelle manière de décrire les connaissances. De ce fait la communication entre le client et le serveur est rendue plus robuste.

La mode féminine est choisie comme domaine d'application pour ce travail parce que c'est un domaine familier comportant des produits avec des caractéristiques communes (tels que la marque, la couleur, etc.). Il y a là un potentiel à fournir un site Web attrayant, auquel beaucoup de clients seraient possiblement sensibles. En outre une évaluation statistique est faite afin de tester la validité de l'approche que nous avons proposée. De nos jours, il y a beaucoup d'applications de commerce électronique sur le Web. Seule, un petit nombre d'entre-elles utilise des techniques de CBR pour la recherche de produit, même si il existe beaucoup de systèmes qui offrent des fonctionnalités pour personnaliser les produits. Notre approche dans ce mémoire permet la détermination efficace des produits qui répondent aux exigences de l'utilisateur.

Un article [Ma & Aïmeur, 2001] basé sur ce travail a été accepté par IEEE Computer Society Press dans les Proceedings of the Tenth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Research, MIT, Cambridge, Massachusetts.

Abstract

E-commerce is a rapidly growing area on the Internet. There are various types of online shopping sites that help consumers find products to purchase. The intelligent agent is among the most promising technologies used in building e-commerce systems. *CBR (Case-Based Reasoning)* is a highly effective mechanism for developing intelligent agents. In this thesis, we propose an intelligent agent based on CBR, called XMLFinder, which recommends products to online customers.

There are two main characteristics in this proposed agent system. The first characteristic which is based on CBR technology is concerned with the way in which this system enables the consumers to weight, score and adapt their product requirements. Usually, similarity based results (close enough results) are more important than getting results with an exact match, because most online shoppers can't give an exact query to describe their requirements. A new notion, called *the deficiency*, is introduced. The deficiency of an attribute measures to what degree the system considers this attribute to be susceptible to a user's preference. The proposed heuristic can then be expressed as such: if the client does not choose the most deficient of the considered cases, then the importance of the most deficient aspects must decrease and the importance of the preferable aspect must increase. The intuitive justification behind this heuristic is that clients have a tendency to first choose his preferable aspect to which they attach more importance. This is called short-term profiling in this thesis. Also the combination with long-term profiling such as the consumers' constraints makes our system provide recommendations more efficiently.

The second characteristic is the use of *XML (eXtensible Markup Language)*. XML is a useful communication language enabling large packets of formatted information to be exchanged thereby reducing network traffic. XMLFinder is constructed on a three-tier net architecture, in which XML documents take a key role in each tier. As a possible replacement to HTML, XML helps the web support intelligent application. *XSL (eXtensible Stylesheet Language)* is used to provide a clean separation between the content itself (XML) and the specific format required by a target application or output document. So a wrapper is provided for the catalogue in XML file translated by XSL. The cases and similarities' representation in XML is a new way to describe the knowledge. This makes the communication between client and server more robust.

The women fashion is chosen as an application domain for this thesis because it is a common product with common characteristics (such as brand, color, etc.). There is a potential to provide an attractive web site, which most people can relate to. Further more a statistical evaluation is made in order to test the validity of the approach we proposed.

Nowadays, there're many electronic commerce applications on the Web. But only a very small number of these applications use CBR techniques for product retrieval, apart from that fact, there are not many systems, which offer possibilities to customize the products. Our approach in this thesis allows efficient determination of products that meet the user's requirements.

A paper [Ma & Aïmeur, 2001] about this work has been accepted by IEEE Computer Society Press in the proceedings of the Tenth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Research, Knowledge Media Networking Workshop, MIT.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electronic commerce [Adam *et al*, 1999] [CommerceNet] offers a huge variety of application areas nowadays. Much effort is put into research and exploration of these areas. The online shopping sales support is one of the most important problems. The turn of Internet to a big electronic marketplace has given rise to a lot of difficulties. Finding relevant information offered in a huge search-space and coping with the many different user interfaces can cause significant problems for a human being. Even when the problems of finding relevant information are solved, we still face the issue of the non-uniform way of presentation. The terms "lost in hyperspace" and "information-overload" have been widely used to describe these problems. The best thing to do would be to instruct an intelligent assistant -- *intelligent agent,* to provide some product recommendation. This thesis is concerned with this problem. We construct a shopping agent model – XMLFinder by using modern technologies to make consumers free from the information overload and to make their decision-making easier when shopping online.

Intelligent agents [Nwana & Ndumu, 1998] [Hayzelden & Bigham, 1999] are software programs that independently perform tasks on behalf of a user in a network. It is different from current programs in three ways. They are *proactive* – initiate actions; they are *adaptive* -- learn from the user's preference and habits and they are *personalized* -- change their way of helping the user according to what they have learned from users [Maes, 1994a]. *Electronic shopping agents* are agents that assist in searching for product items on behalf of a user. Upon receiving a request, the agent searches products database or relevant online shops throughout the Internet for items that match the search criteria. The agent would return to the user a detailed description of the items, the price of the items and a direct link to the virtual store.

Case-based Reasoning (CBR) [Aamodt & Plaza, 1994] [Andrew_Broad] [AI-cbr.org] is a promising technology and it is becoming highly effective mechanism for developing intelligent agents. The main idea of the CBR is to use the solution of a problem that has been solved earlier in order to solve a new problem. The obvious cases are descriptions of the commodities on sales and the task is to identify the case configuration that meets the user's requirements. XMLFinder utilizes CBR technology as its intelligent mechanism to provide the products' recommendations to consumers. The detailed techniques of CBR would be discussed in this thesis.

There are some shopping agents that already exist on the Internet. The problem is that most agents need to do a lot of maintenance to translate data from different sources to a well-known format. As a result, in many instances browsers can't get data automatically. They have to first collect information and then assemble it in a useful format. A better way for vendors to publish their information would be through the use of a well-known data format. *XML (eXtensible Mark-up Language)* [W3.org] [XMLS.com] [XEDI.org] standard meets most of the required criteria and provides a promising tool in the development of this critical area of Internet development. It is an excellent format for agent interaction. Because XML is tagged, the agents expecting only well formed XML format can be extremely fault-tolerant, ignoring tags they don't understand. XMLFinder is constructed on three-tier net architecture, in which XML takes an important role. Each tier has its own responsibilities; together they form a cohesive, flexible, and scalable shopping application.

1.1 Objective and scope of this thesis

The objective of this research is to construct an intelligent agent (XMLFinder) to meet the consumers' needs when shopping on-line in order to reduce "information overload". This thesis would focus on the following issues:

1. Study the user's shopping behaviors in general, and particularly in business-to-consumer e-commerce applications so that we can employ agent technologies in different stages of the user's purchases.
2. Construct shopping agents network architecture – XMLFinder in order to meet the usability, interoperability, and dynamic content requirement of e-commerce. And provide a well-known data format on solving the problems on data interchanging and data presentation.
3. Adopt CBR technology as the recommendation mechanism, which is focused on the similarity measure, weight modification, case representation and adaptation. And explore the ways of building a user model (user profiling) to acquire the knowledge, because software agents need the specific knowledge from users to assist them in achieving their goals.
4. Construct a Fashion agent on the web as a prototype of XMLFinder. And an experiment and evaluation are made in order to test the validity of the approach we proposed.

1.2 Organization of this thesis

This thesis is divided into eight chapters. Its content is organized as follows:

Chapter 2 provides a literature review of the intelligent agent in 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. Several sample agent-based systems are introduced and their underlying techniques are analyzed.

Chapter 3 introduces the related knowledge and technologies used in our system. XMLFinder system is a synthesis of many concepts; it fuses various latest technologies, tools and protocols. CBR technologies are introduced as the agent's intelligent mechanism; XML is used as the communication foundation; *JSP (JavaServer Page)* [SUN_JSP] is described as web server to support web applications, which also provides the support to XML documents. Also an overview of e-commerce and agents technologies is presented.

Chapter 4 describes the three-tier network architecture of XMLFinder, in which XML takes a key role in each tier. In middle tier, it presents a multi-agent -- a Recommend-Agent and an XMLAgent. The former gets personalized content from client browsers, and then runs a task on behalf of the user. The latter is responsible for the generation, integration and publishing of XML documents.

Chapter 5 presents the detailed methodologies and algorithms, which is based on a new e-commerce *CBR-cycle* proposed in [Wilke *et al*, 1998], which is modified from the traditional CBR cycle [Aamodt & Plaza, 1994]. We adopt a metric in the context of the nearest neighbor method, and then use this cycle combined with a heuristic we devised to create the users' short-term and long-term profiles. Also the similarity measure, case representation and adaptation are discussed in this chapter.

Chapter 6 presents a Fashion Agent to explain how the prototype of XMLFinder architecture and methodologies function. We introduce the application domain, the running environment, and implementation. In this chapter, we will also explore other domains in addition to the clothes domain.

Chapter 7 describes an experiment and evaluation of this system, which is based on the collection of the user's feedback. Two methods are used in evaluation. One is using the normalized Euclidean distance to measure the similarities between the recommended results and its re-rank orders provided by testers. Another is comparing the rating results to find the whole performance of the agent in general.

Chapter 8 concludes with the features of XMLFinder as an intelligent agent, and the contributions of this thesis. And then discusses its future work. A paper [Ma & Aïmeur, 2001] about this work has been accepted by IEEE Computer Society Press in the proceedings of the Tenth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Research, MIT, Cambridge, Massachusetts.

Chapter 2

State of the Art -- Agent-based Systems in E-Commerce

The Internet will revolutionize the way people buy products in the future. Intelligent agents – software programs that will attempt to deliver what a network or database user is looking for-- it has been a major focus of computer research labs for several years. The popularity of the World Wide Web has created the most fertile ground ever for commercial application of agent technology -- shopping agents or "bots", which provide recommendations to users on the web.

2.1 Consumer Buying Behavior model

Stemming from the traditional marketing Consumer Buying Behavior (CBB), [Guttman *et al*, 1998] divide the electronic marketplace into six stages: Needs identification, Product brokering, Merchant brokering, Negotiation, 'Purchase and Delivery' and 'Services and Evaluation' (Figure 2.1).

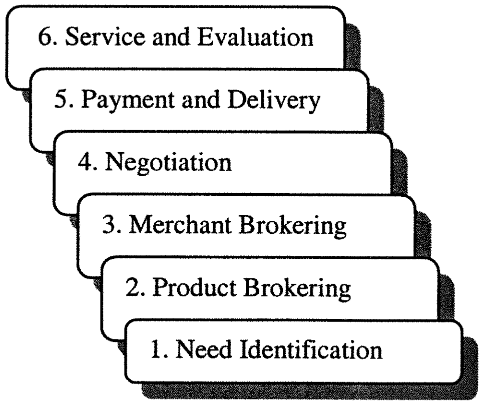


Figure 2.1 Consumer Buying Behavior Model

Although CBB research focuses on many areas, it has some limitations. For example, CBB research focuses primarily on retail markets, and even in retail market, not all shopping behaviors are captured (e.g. impulse purchasing), still, it is a wonderful tool to understand the roles of agents as mediators in electronic commerce.

- Need identification

This stage characterizes the consumer becoming aware of some unmet need, and to stimulate consumers through product information. For example, the need can be achieved by added functions like advertisement features, email and similar notification services, which will keep the consumer up to date with new products and goods available on the market. Generally, the information corresponds with users' profile and their last deals.

- Product Brokering

This stage is responsible for the retrieval of information to help the consumers to determine what to buy based on criteria they provide. For example the consumer provides a set of specific criteria for a desired product, and then the agent will give some answers according to the set of the user's preferences.

- Merchant Brokering

This stage is based on the merchant's information and provides the evaluation of merchant alternatives according to consumer-selected criteria. Combining with product brokering stage, it will help consumers to determine who to buy from.

- Negotiation

Negotiation [Wilke *et al*, 1998] is an important part of the selling process via the Internet, and in order to support customers in a sufficient way, electronic commerce system needs the ability to negotiate. There are different protocols when negotiating, such as cooperative negotiation, classified negotiation, stock market negotiation, retail auction negotiation, cooperative negotiation, etc [Guttman & Maes, 1998].

- Purchase and Delivery

This stage can either signal the termination of the negotiation or occur sometime afterwards in cases where it includes the payment transaction using online payment systems. The actual online delivery of the products is possible if the good is in electronic form, but physical goods have to be shipped by the traditional way by the merchants. In some cases, the available payment or delivery options can influence product and merchant brokering.

- Product service and Evaluation

This stage involves product service, customer service, and an evaluation of the satisfaction of the overall buying experience and decision. Agent-based distributed trust and reputation mechanisms enable customers to share and combine their experiences and use merchant and product reputations as additional aspects of brokering and negotiation. For example the system can include some special deal for specific agents, enabling all sorts of special offers prize cut downs for handling either long time or first time customers.

2.2 Agents' roles in e-commerce systems

In CBB model, these stages represent an approximate simplification of complex behaviors. From this CBB perspective, we can identify the roles for agents as mediators in electronic commerce. We will describe some agent-based e-commerce systems, and then compare them in the main stages of CBB model namely product brokering, merchant brokering, and negotiation stages. Table 2.1 lists the three main stages and which category each system falls in.

Table 2.1 Roles of some Agent-based Systems in E-Commerce

	Persona Logic	FireFly	Virtual Letting Agent	Selecting Mobile Phone	HVAC	Bargain Finder	MySimon	Auction Bot	Tete-a -Tete
Product Brokering	X	X	X	X	X		X		X
Merchant Brokering						X	X		X
Negotiation								X	X

2.2.1 Product brokering

This stage will allow consumers to determine what to buy by searching/retrieving some product information in order to meet their requirement. As shown in table 2.1, all systems except for AuctionBot [Auctionbot.com] and Bargain Finder [BargainFinder] fall in this category. And most of commercial shopping agents are in this category.

PersonaLogic [Personalogic.com]

"What college? Which car? What pet? ... Why not take the head scratching out of decision-making and use one of our guides? They're easy-to-use and highly informative." This is the foreword of PersonaLogic system. The Web shopping system can filter out the unwanted product according to the needs of consumers by specifying their constraints. PersonaLogic is marketed as a service that merchants offer to their customers, which enables them to select the best product from merchants' catalog.

Firefly [Firefly.com]

Firefly has created agent system software called Firefly Password. They hope that the software will become the standard in shopping agent software. The Firefly Passport is free to users who agree to fill out surveys and rank Web site as they surf the Internet. Based on the data collected a customer profile is created, and a process of collaborative filtering is used. The web site operators can then match users with similar profiles and make recommendations based on users' shared interests. This system is mainly used to recommend commodity products such as music and books.

WEBSELL [Cunningham, 2000]

WEBSELL is a project done by four commercial companies and two universities. They have developed an intelligent sales support technology that supports Web shoppers in two aspects of the sales process: "Helping the users identify and articulate their requirements" and " Identifying the products best suited to those requirements." Its modular architecture has several components, which include:

- Representation of product data and knowledge,
- Dialogue with the customer (Pathway),
- Product search through case-based retrieval,
- Personalization and collaborative recommendation,
- Product presentation,
- Product customization

But, none of available demonstration applications exploits all the facilities that WEBSELL offers. Here we introduce two key roles for WEBSELL.

Virtual Letting Agent

It is a system for finding apartments for renting, it helps the users identify and "articulate" their requirements and find products best suited to those requirements. The requirement elicitation process is straightforward. The user's input is captured in a single screen, the set of requirements are passed to the case-base product retrieval engine on the server and the most suitable apartments available are returned.

Selecting a Mobile Phone [Cykopahts.com]

The application is for helping users to select a mobile phone. In this scenario, expressing the user's requirement is not so straightforward. It is a classic application where WEBSELL can assist because there is a difficulty in defining the user's requirements and in mapping those requirements to available phones and connection packages. Decision tree technique is used to manage a dialog and to identify a requirement description.

HVAC

HVAC [Watson & Gardingen, 1999] is an air conditioning sales support system of Western Air (a distributor of air conditioning system) in Australia. It helps the sales staff to be able to give fast accurate estimates to prospective customers on the spot, so that it can avoid a danger that the less knowledgeable sales staff might give technically incorrect quotes. The system uses a web site that sales staff could access from anywhere in the country. Through a forms interface the prospects requirements could be entered into the system. Then they would be passed to a CBR system that could search the library of similar and past installations.

2.2.2 Merchant brokering

Whereas the product brokering stage compares product information, the Merchant brokering stage compares the information provided by merchants, such as price, or other added services.

BargainFinder [BargainFinder]

It was the first shopping agent for on-line price comparison. Just type in the artist and album name of a rock or pops CD, and let the agent get prices from several retailers. At the beginning, merchants always have the option to "block" an agent pinging their sites for data. It's technically easy to screen out traffic from specific IP addresses. Three years ago when BargainFinder launched to get compact disc prices, it found some on-

online music merchants blocking the technology. But now few merchant sites block agents because they know that the price is just only one part of comparison.

MySimon [MySimon.com]

MySimon's Internet e-commerce hub is the largest comparison shopping site on the web with over 2000 merchants, in categories such as computers, book & music, electronic, fashion, flows, sporting goods, toys and many more. MySimon's shopping service offers unbiased information on products and merchants, including price comparison, availability and other merchant information.

Unlike some agents, MySimon doesn't charge merchants to be part of its searches. Instead, its business model is commission -- 2%-5% of each sale made to a buyer who accessed the site through a MySimon search.

2.2.3 Negotiation

Negotiation is the process of determining the price and other terms of a transaction. The negotiation in traditional commerce includes market, fine art auction house, flower auction, and various special haggling. However, the intelligent agents negotiate with each other using sophisticated strategies based on rule-based system or some utility function defined by the individual user.

AuctionBot [Auctionbot.com]

AuctionBot comes in many varieties, depending on the type of goods involved, the mechanism for determining prices and communication protocols. AuctionBot is designed to support a range of auction types and customization parameters, according to the specifications of whoever sets up the auction.

Nowadays, sellers operate most Internet auctions. It creates a potential conflict of interest, providing an incentive to introduce the rules and use bid information so to maximize revenue over time. In contrast, the AuctionBot dose not favor particular individuals or particular outcomes. Rather they are interested in a smoothly run mechanism, with well-defined and well-explained rules, and faithful implementation.

Tete-a-Tete

It is a research project following the widely known Kasbah project in MIT Media lab. It incorporates the idea of value comparison rather than just simple price comparison. The value comparison is achieved with a utility function to maximize the individual user's needs, using a multi-attribute utility theory (MAUT). It negotiates across multiple terms of a transaction such as warranty length and options, shipping time and cost, service contract, return policy, etc. It integrates all three stages in CBB model. But Tete-a-Tete dose not offer auction mechanism or payment and delivery functionality and also lacks brokering functions.

2.3 Technologies used in agent-based e-commerce systems

The majority of agent-based e-commerce systems are developed using content-based, collaborative-based, or constraint-based filtering methods. Recently, case-based reasoning is becoming a promising technology in e-commerce agent systems. In this section, we will discuss these technologies and the related agent systems. Table 2.2 lists the main technologies and their associated providers.

Table 2.2 Technologies' Comparison of some Agent-based Systems in E-commerce

	Persona Logic	FireFly	Virtual Letting Agent	Selecting Mobile Phone	HVAC	Bargain Finder	MySimon	Auction Bot	Tete -a- Tete
Content-Based						X		X	
Collaborative-based		X							
Constraint-based	X						X		X
Case-based			X	X	X				

2.3.1 Content-based filtering

Perhaps the most straightforward method used by agent systems is the one that analyzes the objects and compares them with the user's preferences. This is called content-based method since the starting point of the adaptation is the content of the object.

BargainFinder tries to collect information from many different Web information sources. When users specify the details (title and author) of the CD they went to buy, BargainFinder transparently searches multiple online bookshop databases worldwide. Those web information sources intended to be read by humans and their content is rendered according by (i.e. HTML). Different sources have different inputs (e.g. CGI, Applets) and presentation methods, so agent systems have to adjust their interaction methods depending on the Web site.

2.3.2 Collaborative-based filtering

The Collaborative Recommenders recommend products by using collaborative technique. They use feedback and rating from different consumers to filter out irrelevant information. Firefly uses this technology, in the sense that it tries to analyze or understand the features or the descriptions of the products. The ranking of the consumer is used to create a "likability" index for each document. This index is not global, but is computed for each user on the fly by using other users with similar interests.

2.3.3 Constraint-based filtering

Like the content-based approaches, *constraint-based filtering* uses features 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 to be formulated in terms of variables, domains, and constraints. Once formulated in this way, however, a number of general purposes (and powerful) *constraint satisfaction problem* (CSP) techniques can be employed to find a solution [Tsang, 1993].

Many problems can be formulated as a CSP such as scheduling, planning, configuration, and machine vision problems. In PersonaLogic, CSP techniques are used in the Product Brokering CBB stage to evaluate product alternatives. Given a set of constraints on product features, PersonaLogic filters products that don't meet the given "hard" constraints and prioritizes the remaining products using "soft" constraints (which need not be completely satisfied).

Tete-a-Tete uses CSP techniques to assist shoppers in the Product Brokering, Merchant Brokering, and Negotiation CBB stages. Consumers will provide product constraints (as in PersonaLogic) as well as merchant constraints such as price, delivery time, warranty, etc. Hard and soft constraints are used to filter and prioritize products and merchants as well as to construct a multi-attribute utility that are used to negotiate with the merchants. Tete-a-Tete's argumentative style of negotiation resembles a distributed CSP with merchants providing counter-proposals to each customer's critiques.
MySimon's proprietary Virtual Learning Agent (VLA) technology takes the approach that creates mass quantities of intelligent agents that mimic human behavior. MySimon can be "trained" to extract specific information from any merchant Web sites. The agents are created by SPIs (MySimon Product Intelligent) who are non-programmers that interact with the VLA system. The VLA system translates the SPI's Web navigation behavior into a proprietary programming language called "V" to produce mass quantities of agents.

2.3.4 Case-based reasoning

Case-based reasoning is a problem-solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, the central tasks that all case-based reasoning methods have to deal with are to identify the current situation, find a past case similar to the new case, use that case to suggest a solution to the current problem, evaluate the proposed solution, and update the system by learning from this experience [Aamodt & Plaza, 1994]. Retrieval is a huge area of research in CBR. And right now the commercial CBR tools normally offer the two alternatives of k-Nearest Neighbors and Decision Tree [Cunningham, 1998]. It would be content-based or constraint-based. More detail about CBR technologies will be discussed in chapter 3.

WEBSELL is a CBR-based intelligent sales assistant project. The retrieval component of WEBSELL project provides different similarity-based retrieval algorithms such as "complete brute-force search", "case-retrieval nets" and "similarity-based retrieval by approximation with SQL queries." The main source of intelligence in Virtual Letting Agent is the similarity measure that identifies goods matching and the score from this metric is used to rank the apartment returned. Selecting a Mobile Phone uses a technology of WEBSELL, called "Pathways", which allows the requirement elicitation process to be expressed as a tree. The user starts at the root node and is led through a series of questions to elicit a complete requirement description. The output from the navigation of a pathway is the data gathered or computed during the run; the output is in a customizable format, usually XML. In addition to the case-based recommendation approach, WEBSELL also provides a facility for collaborative recommendation based on user profiling. In the context of WEBSELL, user profiles support the potential personalization of all aspects of the sales process.

HVAC uses a nearest neighbor retrieval algorithm, in which a similarity table is used to rank a set of cases. This system is constructed on client-server architecture. On the sales staff (client) a Java applet is used to gather the customer's requirements and send them as XML documents to the server. On the server side another Java applet (a servlet) uses this information to query the database in order to retrieve a set of relevant records. Once retrieval from the database is complete, the records are ranked by a nearest neighbor algorithm and dynamically converted into XML for presentation to the client browser. An XML case representation is used by this system.

BidFind [BidFind]

BidFind is an auction search. Thousands of items indexed daily from popular online auction sites visited by their agents. It collects data from thousands of dynamic pricing related sites on the net, auctions, barter, collectibles, exchanges, etc. BidFind's agent technology services can provide data collection for the companies or businesses.

Bonzi Buddy [bonzi.com]

Bonzi Buddy is an active MS Agent who lives on the browser window. He can surf the web with the users as their sidekick. He can talk, joke, browse, search, e-mail, and download like no other friend the users have ever had. He can keep the users on schedule. He also can track their appointments and tasks with his built-in calendar. He even has the ability to compare prices on the products and help users save money.

BottomDollar [BottomDollar.com]

Bottom Dollar finds the lowest prices on the Web - fast. Just pick one of the categories (books, music, movies, hardware, software and toys) in the title, and then start searching. Prices are always the newest and they are direct from the sellers.

Fido-theShoppingdoggie [Shopfido.com]

Fido provides a frequently updated, centralized database of vendor products and prices with a simple searching mechanism. Fido's database contains 100,000 products from 100 vendors.

<u>One-Click Computer ESP Bargain Agent</u> [Uversion.com]
Computer ESP Agent is JavaScript-Powered. There are four versions: Product Description Agent, Stock Symbol Agent, Full Text Description Agent, and Vendor Part Number Agent.

<u>RoboShopper</u> [RoboShopper.com]
RoboShopper is an add-in to Netscape Navigator and Microsoft Internet Explorer, which makes online shopping fast and easy. Just tell RoboShopper what you are looking for, and it will automatically query online stores and then display the results in a convenient report, so you easily compare and make the best buying decision.

<u>Shopping Explorer</u> [ShoppingExplorer.com]
Shopping Explorer is a software package that allows you to simply tell its shopping agent what you are looking for and it will search the web sites of hundreds of participating online merchants all over the world.

<u>VirtualOutlet.com</u> [VirtualOutlet.com]
They have a nice database of products from different cyber-merchants where you can make a comparison between products and services on the Web.

Chapter 3

Technologies Used in XMLFinder

XMLFinder is a synthesis of many concepts. It fuses various modern technologies, tools and protocols. It has following features:

- Case-based Reasoning (CBR) is used as the intelligent mechanism
- JavaServer Page (JSP) is used as the communication architecture
- XML is used as its "data interchange modeling" layer
- Object-Oriented Java language is used as programming tool, JDBC is used as the data providing layer

Figure 3.1 shows the technologies' architecture of our system.

In this chapter, we overview e-commerce, agent technology and related technologies used in the agent-based e-commerce system.

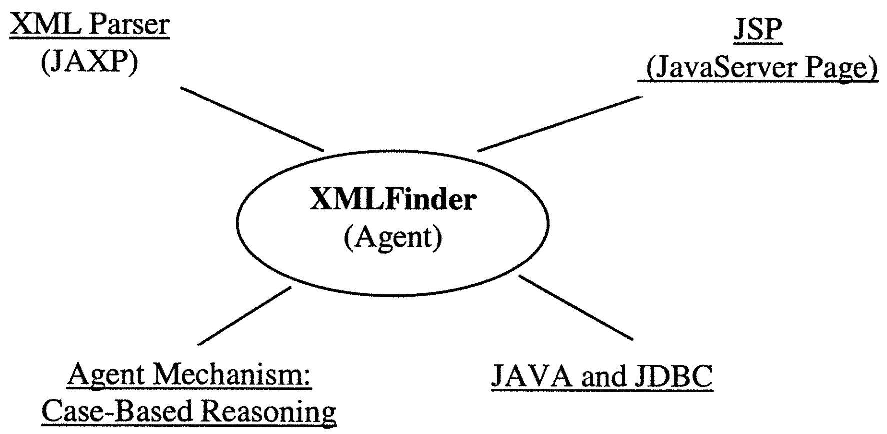


Figure 3.1 Technologies' Architecture of XMLFinder

3.1 Overview of Electronic Commerce

The term *e-commerce* has no widely accepted definition. In a loose sense it means doing business over the Internet, selling goods and services which are delivered offline as well as products that can be "digitized" and delivered online, such as computer software. E-commerce may be defined as "the technology, processing, and operations, which occur when business transactions are done automatically over networks, using IT [OKIgroup]." E-commerce includes transactions between businesses and consumers (B2C), businesses and businesses (B2B), as well as operations internal to companies, the conversion of government procurement to electronic systems, and transactions between consumers. Mostly trades can be among businesses or between businesses and consumers. But the Internet also encompasses a wider spectrum of potential commercial activities and information exchanges. For instance, it offers firms, individuals, and governments an electronic infrastructure, which enables the creation of virtual auction-markets for goods and services where previously they did not exist. EBay.com, for example, was among the first successful sites to provide a framework where consumers can trade a wide diversity of goods and services with each other (consumer to consumer, C2C) and, at least in principle, with businesses (consumer to business, C2B) [Adam *et al*, 1999].

3.1.1 History of e-commerce

It can be argued that e-commerce had its advent upon the completion of the first Trans-Atlantic telegraph cable near the turn of the 19th century. However, modern e-commerce truly began to emerge with the introduction of Electronic Data Interchange (EDI) [Clarke, 1998] in the 1970's and 80's. EDI created many opportunities for business but there were plenty of challenges as well. For instance, EDI offered the ability to exchange data and information very quickly.

Electronic Data Interchange is a safe and secure means of exchanging standardized business forms electronically. Since the late 1970s, it has been used to automate the procurement process and reduce costs. Analysts estimate that business trade is in excess of $150 billion over EDI and obtains savings of 5-10% (US Department of Commerce). However, because EDI is usually run over value-added Networks, the costs are high. For example, the cost to add one trading partner can exceed $50,000 [Bolin, 1998].

While the value of EDI transactions are currently 14 times higher than business-to-business transactions over the Internet, a transition is occurring. The Gartner Group [Gartner.com] predicts that 80% of current EDI users will implement extranets by 2003. This transition will allow small businesses that traditionally conduct business via fax and phone to cost-effectively participate in the procurement process and become the suppliers to larger companies.

However the most serious problem businesses were facing was how this electronic exchange should be formatted so that their respective systems could communicate. In many ways, the same challenge exists today. Other challenges that businesses struggled with included security and accessibility. As a result, early e-commerce was limited to large corporations with deep pockets. The revolutionary impact on commerce will be dramatic, as companies come to realize that e-commerce is more than just EDI. However EDI and Internet-based e-commerce may coexist in a long period.

E-commerce has the added capability to both attract and service customers. This may provide a further, more compelling reason to invest increasing resources in it. The utility of Internet and the Extensible Markup Language (XML) has lowered the entry-level barriers to commerce, in both cost and complexity. XEDI.ORG presents a more direct and effective approach for translating EDI into XML called XEDI (zee-dee) [XMLS.com] [XEDI.org], which allows the integration of XML with the traditional method of EDI.

3.1.2 Statistics on e-commerce

It is difficult to measure how widespread e-commerce is. Two often-cited indicators that can be compared internationally are the numbers of Internet hosts and secure servers. These indicators show e-commerce expanding at a very rapid pace. In March 2000 there were 66,810 secure servers, up 97 percent compared with a year earlier and Internet hosts have increased at exponential rates [Coppel, 2000]. Other indicators of Internet usage, such as the number of web users, web sites, and new domain name registrations also imply rapid growth. But disparity across countries and regions is wide.

The value of e-commerce transactions is more difficult to measure. Few statistical agencies systematically measure electronic transactions, although a number of countries intend, or are in the process of developing indicators related to electronic business processes. A number of consulting groups, however, have published estimates of e-commerce transactions. Those vary widely, due to diverse definitions and scope. Nevertheless, taken together they all reveal an extremely rapid growth of e-commerce.

With the majority of growth occurring in the business-to-business category, IDC [IDC.com] predicts that online sales will total $900 billion between 1999 and 2002. Business-to-Business e-commerce will rise from a predicate $131 billion in 1999 to $1.5 trillion in 2003 (Forrester Research [Forrester.com]). The forecast for e-commerce is expressed by the following statement: " By2003, the Internet will become the predominant mechanism for conducting business -- either to consumers or between businesses," according to a Gartner Group analyst.

The number of electronic consumers in the US is a total of 41.20 millions in 2000 – over double their number of 21.10 millions in 1998, and it is predicted that the number of purchasers would be 72.10 million in 2003 (Donaldson, Lufkin & Jenrette) [DLJ.com].

The online purchase revenues are $45 billion in 2000, much more than the $8 billion in 1998, and it is predicted to be $300 billion in 2004 [KeenanVision.com]. Business-to-business spending over the Internet increases from $30 billion in 1998 to $100 billion in 2000, and to approximately 1.0 trillion in 2004 [KeenanVision.com]. "By 2004, most corporations will start becoming enterprise service providers, and will begin managing Internet access as an integral part of the company networking strategy, with both Intranet and extranet business services [Gartner.com]."

3.1.3 Factors stimulating and limiting growth in e-commerce

Several factors will contribute to e-commerce growth. On the whole, these factors include lower purchasing costs, reduced and more accurate inventory, lower cycle times, improved customer services, lower sales and marketing costs, and new sales opportunities.

Stimulating factors:

- Reduced and more accurate inventory
- Lower cycle times
- Improved customer service
- Lower sales and marketing costs

While the Internet and e-commerce bring extensive benefits to organizations and their customers, they also face challenges that may limit growth. Two major technical factors likely to influence the future expansion of e-commerce are the extent to which IT companies can invest in network capacity and the speed of data transmission. As regarding technical problems, consumers have concerns over privacy, consumer protection, and security of credit card purchases, order fulfillment, and delivery.

Limiting factors [Adam *et al*, 1999][Coppel, 2000] would be:

- Lack of a predictable legal structure
- Unclear content regulation

- Insufficiently defined government involvement
- Lack of privacy protection (this inhibits growth)

E-commerce still continues to develop rapidly and it could have profound impacts on the economy and technology of this world.

3.1.4 Technical support of e-commerce

E-commerce systems require a technical architecture that is scalable, enables resource sharing, is based on asymmetrical protocols, and provides transparency of data locations and decision support. It also supports multimedia information, allows mix-and-match of various software and hardware platforms, encapsulates services, support communications through message based exchanges, and guarantees security and integrity of data. Central to providing such features is a standardized telecommunication infrastructure with sufficient bandwidth for efficient transmission of text, voice, image, and voice data. We will discuss the networking architecture of XMLFinder in chapter 4.

One important technology component in the world of e-commerce is interoperability, which requires distributed object management and agent software that can help users in solving business problems. The emerging of XML in conjunction with XSL injects a fresh blood into e-commerce, which is as a powerful yet standard mechanism for dealing with disparate application data formats, and providing a sound, flexible data integration solutions [XML/XSL, 1999]. We will discuss XML in the following section. Some of the related technologies used recently in e-commerce are:

- Telecommunication infrastructure (J2EE)
- Decision support system (data warehouse, data mining, etc)
- Interoperability (CORBA, intelligent agent, etc)
- Markup Languages (HTML, XML)

This thesis concentrates on intelligent agents, Case-based reasoning and XML. They are described in the following sections.

3.2 Agents technology

The Internet provides us with enormous amounts of information, and it is involving more and more people in its world of bits. Computers are becoming essential tools for entertainment, socializing, and obtaining information or news. Improvements in computer technology allow us to have access to information from anywhere at anytime. People need help in dealing with this information overload to separate useful information from noise. An intelligent agent is used to perform that function.

3.2.1 Intelligent agent and its properties

There is no widely accepted definition for, or approach to, agents. Intelligent agents, called also software agents, are a new category of information-society tools. Generally, the term refers to software programs that independently perform tasks on behalf of a user in a network. Intelligent agents are different from current programs in three ways [Maes, 1994b]:

1. they are proactive – initiate actions,
2. they are adaptive -- learn from the user's preference and habits,
3. they are personalized -- change their way of helping the user according to what they have learned from users.

Agent terminology is used to refer to a wide range of solutions being proposed by different organizations for different purposes. One view of intelligent agents in [Nwana & Ndumu, 1998] is that they use three intelligent attributes to derive three types of agents: collaborative agents, interface agents, and truly smart agent (as shown in Figure 3.2)

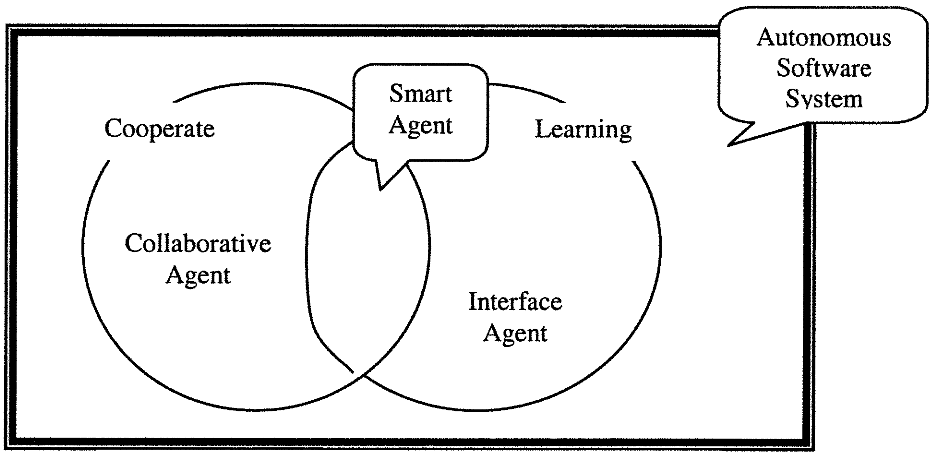


Figure 3.2 Agent Classification

Agents are also classified as information agents, task agents, and interface agent in [Maes, 1994b]. Task agents help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents; information agents provide intelligent access to a heterogeneous collection of information sources; interface agents interact with users, providing a mechanism whereby users can specify tasks, and inspect the results, they may acquire, model, and utilize user preferences to guide system coordination in support of the user's tasks. The table below summarizes some of the attributes of agents that researchers have highlighted.

Table 3.1 Properties that Software Agents can Exhibit [Hayzelden & Bigham, 1999]

Property	Description
Social ability	A software agent is able to use communication as a basis to signal interest or information to either homogeneous or heterogeneous agents that constitute a part of its environment. The agents may work toward a single global goal or separate individual goals.
Autonomy	Agents should operate without the intervention of external elements (other agents or humans). They have some kind of control over their actions and internal states.
Reactivity	Agents perceive their environment and respond in a timely fashion to changes that may occur.
Adaptability	Agents are characterized by their flexibility, adaptation, and facility to

	set up their own goals based on their implicit purpose (interest). One of the major characteristics of agents is their ability to acquire and process information about the situation, both spatially and temporally.
Agent Granularity Degrees	Agents may have degree of complexity; Most simple agents are characterized by the lack of intelligent behavior. More complex agents are characterized by their ability to know their environment, to act on themselves and on the environment; their observed behavior is a consequence of their perception knowledge, and interactions.
Learning	Either the agency itself may perform some learning ability or each individual agent may be embedded with a learning algorithm. Learning often allows the agent to alter its future action sequences and behavior such that future mistakes can be alleviated. Learning is often a factor that provides an agent's ability to demonstrate adaptive behavior.
Pro-activity	Agents should exhibit goal directed behavior such that their performed actions cause beneficial changes to the environment. This capability often requires the agent to anticipate future situations rather than just simply responding to changes within their environment.

3.2.2 Electronic commerce agents

Electronic shopping agents are agents that assist in searching Internet for product items on behalf of a user. Users interact with a shopping agent by submitting agent requests. Upon receiving a request, the agent searches products database or relevant online shops throughout the Internet for items that match the search criteria. The agent would return to the user a detailed description of the items found, the price of the items and a direct link to the virtual store where the user can purchase the items. The agent would format the information in a manner facilitating comparison-shopping by the user. The items returned might be sorted by price for example. A shopping agent is one type of interface agents.

[Maes, 1994b] concludes that user interface agents can assist users in a range of different tasks including:

- They perform tasks on the user's behalf
- They can train or teach the user

- They help different users collaborate
- They monitor events and procedures

An agent is different from a search engine. At first glance, search engines such as Yahoo, Lycos, and WebCrawler seem to match some of the capabilities of basic agents. The key difference is that an agent is more interactive and can accomplish many tasks at many different locations. For example, searching on a keyword using a search engine such as Yahoo, people will get a list of matches. They can then follow those links and possibly get their information. If an agent is used, on the other hand, the agent could submit the keywords to many search engines, follow the corresponding links, and gather the information, all without any intervention from the user. It should be noted, however, that the search engines have much aided web users over the past years, yet it is time for the evolution of agents in the searching field.

E-commerce is a rapidly growing area for agents. Consumers are looking for products and services on the Internet, meanwhile, suppliers are looking for buyers to increase market share. The vast amount of information on the Internet causes a great deal of problems for both consumers and sellers. Without shopping agents, shopping on the Internet is a game of chance. Shopping agents can go searching for products for consumers by using things such as store locators, brand locators, category locators, and product locators. They will also be able to query the user's opinions on certain products. In short, the consumer will need to make certain specifications and the agent will then find products that match these specifics.

3.3 Case-Based Reasoning technology

Over the last few years, Case-Based Reasoning (CBR) [Andrew_Broad] [AI-cbr.org] has grown from a rather specific and isolated research area to a field of widespread interest. Activities are rapidly growing - as seen by the increased rate of research papers, availability of commercial products, and also reports on applications in regular use. Especially, with the development of Internet, CBR technologies have been used in Electronic Commerce broadly.

3.3.1 The concept of CBR

The case-based reasoning is an area of exploration within Artificial Intelligence (AI). Studies of case-based reasoning begin with notions about how humans perceive the world and work out problems. A case-based reasoning system simulates these human methods in modeling real world problems. The premise of case-based reasoning is that once a problem has been solved, it often moves efficiently to solve the next similar problem by starting from the old solution rather than by rerunning all the reasoning that was necessary in the first time. So the main idea of the case-based reasoning is to use the solution of a problem that has been solved earlier in order to solve a new problem [Janet, 1993]

A case-based reasoner is the key component of the CBR system. It has the ability to learn from its past experiences. It requires feedback from the objects it applied to. So it can interpret what was right and what was wrong with its solutions. Without feedback, the reasoner might get faster at solving problems but would repeat its mistakes and never increase its capabilities. So in the case-based reasoning, the evaluation and repair are very important.

A case is the most basic element that represents an experienced situation. The case-based reasoning emphasizes the use of concrete instance over abstract operators. It also emphasizes the manipulation of cases over composition, decomposition, and re-composition processes. As a result, the use of more specific knowledge is more efficient than the use of less specific knowledge.

3.3.2 The principle techniques in CBR

The techniques that make up the CBR are case representation, indexing, retrieval techniques, and adaptation. The case-based reasoning is a cyclic and integrated problem-solving paradigm, which utilizes the specific knowledge of previously experienced, concrete problem situations (cases) to find a similar past case, and reuse its solution in the new problem situation, otherwise to solve a new problem.

- Case representation

The representation problem in CBR is primarily the problem of deciding what to store in a case, finding an appropriate structure for describing case contents, and deciding how the case memory should be organized and indexed for effective retrieval and reuse. An additional problem is how to integrate the case memory structure into a model of general domain knowledge, to the extent that such knowledge is incorporated.

The case storage is an important aspect in designing an efficient CBR system, which has tight relationship with case representation. The case-base should be organized into a manageable structure that supports the efficient search and retrieval methods. The designer should balance between the richness of cases' preservation and the simplification of the access and retrieval of relevant cases.

- Indexing

The idea of indexing is central to the representation, storage and retrieval of cases. Each case is tagged with a set of indexes, which serve to guide the search process for retrieving cases appropriate to the current problem situation. Indexing avoids a linear search through the entire case library, which can often contain thousands of cases!
A good index is one that is distinct but not unique, so that a small set of cases will be retrieved [Riesbeck & Schank, 1989]. The more specialized a case is, the larger its set of indexes will be.

- Retrieval

In retrieval, matching or similarity-assessment is digging deeper to realize the similar problem by using indexing vocabulary. People use CBR to build decision-aiding systems, which can retrieve cases better. Case retrieval includes two steps, one is recalling previous cases: retrieve "good" cases that can support reasoning come in the next step. Another is selecting the best subset: select the most promising case(s) to reason with those generated from step one. Recently, two general ways to do cases retrieval are the nearest neighbor retrieval, and inductive retrieval.

- Adaptation

Old solutions are used as inspiration for solving new problems. If they are not exactly matched, old solutions must be fixed to fit new situations, this is the adaptation.
There are four general methods for adaptation:

  - Substitution: substitute values appropriate for the new situation for values in the old solution.
  - Transformation: transform an old solution into one that will work in a new situation.
  - Special-purpose adaptation and repair: used to carry out domain-specific and structure-modifying adaptations not covered by the above methods.
  - Derivational replay: reuses the method for deriving an old solution or solution piece to derive a solution in a new situation.

3.3.3 CBR in e-commerce system

Aamodt and Plaza [Aamodt & Plaza, 1994] established CBR-cycle, which was described by the following four processes (Figure 3.3):

- Retrieve the most similar case or cases
- Reuse the information and knowledge in that case to solve the problem
- Revise the proposed solution
- Retain the parts of this experience likely to be useful for future problem solving

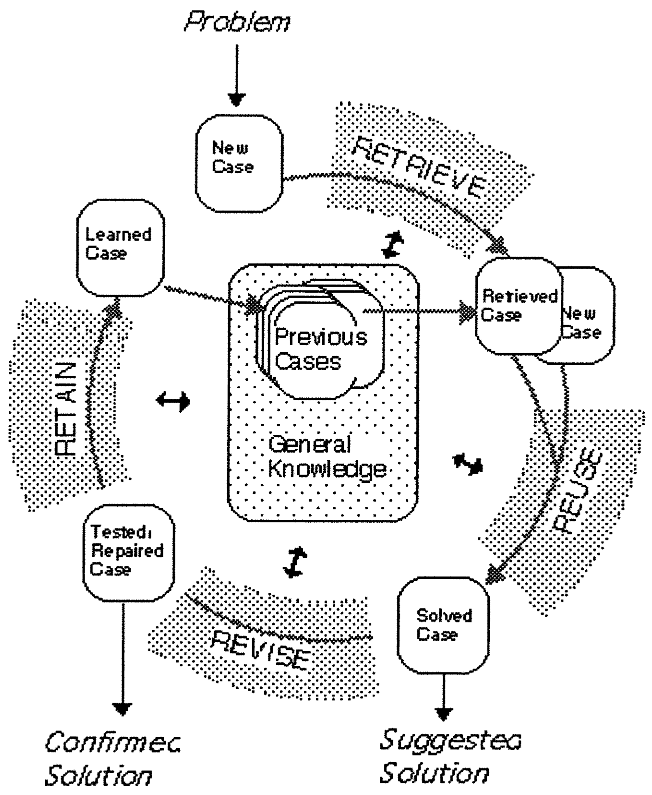


Figure 3.3 Traditional CBR cycle

The development trends of CBR methods can be associated around four main topics [Aamodt & Plaza, 1994]: Integration with other learning methods, integration with other reasoning components, incorporation into massive parallel processing, and method advances by focusing on new cognitive aspects.

[Wilke *et al*, 1998] modified the cycle in their sales support application as shown in Figure 3.4. First, the sales process starts with a set of initial demands stated by the user followed by a *retrieval* process for similar products in the product base of all available products is performed. Second, the retrieved products may be *reused* in a product configuration phase. In this phase, the products are tailored to the specific demands of the user. These modified products are offered to the customer. Third, the user evaluates these offers during the *revise* phase, which results in a set of evaluated products. The customer can state that he accepts certain products or parts of the products or he may state that something is not appropriate. Finally, a new step called *refine* is introduced. This step dose not occur in the traditional CBR cycle. Here, the current user demands are refined based on the evaluations given by the customer. This is required if the current demands cannot be fulfilled on the basis of the available products. After this refine phase the e-commerce cycle is re-entered. Here, a retain phase must not happen because a successful selling of a product dose not lead to an additional product in the product base. The user demands accompany the products from the product base during all phases. This cycle must possibly be re-entered several times until a sufficient negotiation result (meeting of demands and offered product) is archived.

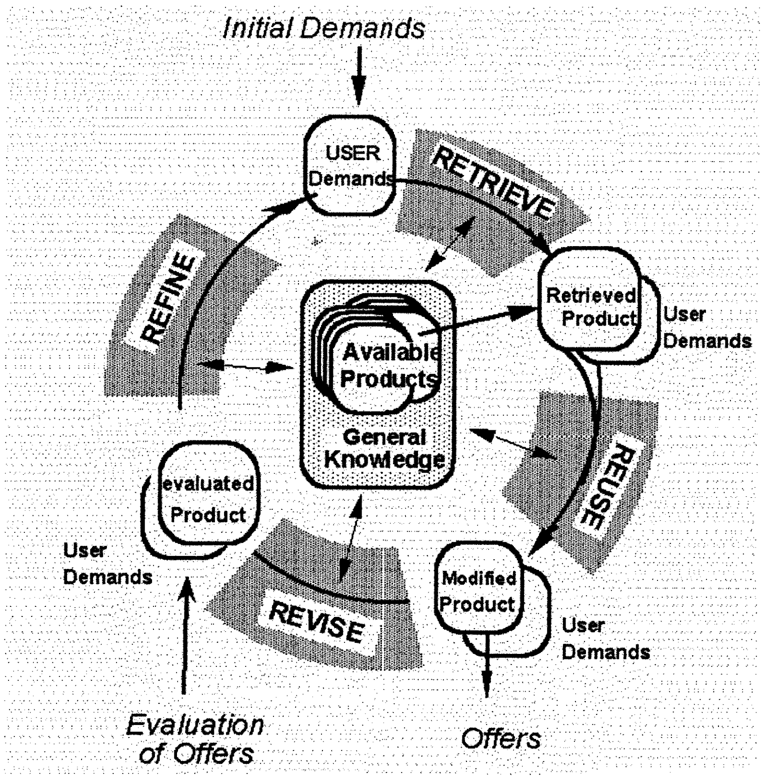


Figure 3.4 The CBR Cycle for Electronic Sales Support Applications

By looking at the trends of CBR application, we find out several applications in e-commerce [AI_EC]. These systems may open up for a more general coupling of CBR - and AI in general - to information systems. The use of cases for human browsing and decision-making is also likely to lead to an increased interest in intelligent computer -- aided learning, training, and teaching. The growing amount of ongoing CBR research in e-commerce has the potential to lead to significant breakthroughs in AI methods and applications.

3.4 XML technology

The web is changing every day. It is not only a place for beautiful web pages, but also a communication medium where businesses can perform daily routines. Therefore the buzz of e-commerce is everywhere. E-commerce is not about creating a shopping center on the web. Rather it is about how a business can communicate with its partners, customers and suppliers. Although HTML is a widespread standard for the presentation of simple documents, it does not disclose anything about either the content or context of the data. By contrast, XML delivers information that separates content from presentation. XML provides the recipient of the data with a business context that allows the recipient to act intelligently on the data.

3.4.1 The concept of XML

"XML is the e**X**tensible **M**arkup **L**anguage, a proposed specification from the World Wide Web Consortium that takes a giant leap beyond HTML [W3.org]." XML will make documents multidimensional, capable of being processed by different programs, delivered by different methods and viewed differently by different users. XML data can further describe itself by including a document type definition (DTD). The DTD defines the structure of the data object up front so that applications can ascertain what attributes is included and what fields are optional. This enables application servers to interpret data on the fly and respond dynamically.

The main point of XML is that, by defining their own markup language, authors can encode the information of their documents much more precisely than with HTML. This means that the programs that process these documents can "understand" them much better, and then process the information in ways that are impossible with HTML.
XML is not HTML. Despite the obvious superficial similarities, the basic HTML behaviors, such as the idea that <img src="bigben.jpg"> includes an image, or that <p> starts a paragraph, are not built in to XML. Early generalizations of XML have led many people believe that XML is just a method for extending HTML by adding new tags. In fact XML and HTML exist in entirely different layers of markup technology. HTML is a tag language (more formally, a markup language) – a set of standard delimiters with standardized meanings that can be put into documents in order to indicate the

role of particular pieces of the document. For example, anything between <H2> and </H2> in an HTML document is understood to be a second-level document head. However, XML is SGML (Standard Generalized Markup Language, ISO 8879), which is same with HTML in this point. But XML is only a fairly small subset of SGML, which retains the key SGML advantages of extensibility, structure, and validation in a language that is designed to be vastly easier to learn, use and implement than full SGML.

3.4.2 Properties of XML

XML provides a number of compelling benefits to developers and users. Also it brings so much power and flexibility to Web-based applications [MSN–XML]:

1. Searching with meaning

   The use of tag in XML document makes the data meaningful. It allows customers to search by features. Without XML, it is necessary for the searching application to understand the schema of each database, which describes how it is built. This is virtually impossible because every database describes its data differently. With XML, however, books, for example, could be easily categorized in a standard way by author, title, ISBN number, or other criteria. Agents could then search these identified bookstore sites in a consistent way.

2. Developing flexible web applications

   Text-based XML document is easy to be delivered to other applications, objects, or middle-tier servers for further processing. Or it can be delivered to the desktop for viewing in a browser. XML, together with HTML has the ability for display, scripting for logic and a common object model that provides the technologies needed for flexible three-tier Web application development.

   - Data integration from disparate sources

     The ability to search multiple, incompatible databases is virtually impossible today. XML enables structured data from different sources to be easily combined.

Software agents can be used to integrate data on a middle-tier server from back-end databases and other applications. This data can then be delivered to clients or other servers for further aggregation, processing, and distribution.
The extensibility and flexibility of XML allow it to describe data contained in a wide variety of heterogeneous applications, from describing collections of Web pages to data records. Again, since XML-based data is self-describing, data can be exchanged and processed without having a built-in description of the incoming data.

- Local computation and manipulation

  After being delivered to the client, data in XML format can be parsed and locally edited and manipulated, with computations performed by client applications. Users can manipulate data in various ways, rather than merely presenting it. Data computations can be performed without additional return trips to the server.

  Separating the user interface that views data from data itself makes the applications more powerful. XML technology can create a simple, flexible and open format to satisfy the web application. However, it is performed only on high-end database before.

- Multiple representation of data

  Once data has been delivered to the desktop, it can be represented in different ways. By describing structured data in a simple, open, robust, and extensible manner, XML complements HTML, which is widely used to describe user interfaces. Again, while HTML describes the appearance of data, XML describes data itself. Since the display is now separated from data, it results in data being presented appropriately. Otherwise local data can be presented dynamically in a manner determined by client configuration, user preference, or other criteria.

CSS and XSL provide declarative mechanisms for describing a particular view of the data.

3. Delivering data on the Web

   Because XML is an open text-based format, it can be delivered using HTTP in the same way that HTML can do today without any changes to existing networks.
   Because XML completely separates the markup notion from its intended display, authors can embed the procedural descriptions in different structure, so that they can show the document in different data views. This is an incredibly powerful mechanism for offloading as much user interaction as possible to the client computer while reducing server traffic and browser response times. In addition, XML lets individual pieces of data be altered with only an update notice, greatly enhancing server scalability as a result of a far lower workload.
   XML compresses extremely well due to the repetitive nature of the tags used to describe data structure. It is necessary to compress XML data. XML can use the compression standard in HTTP 1.1 servers and clients.

3.4.3 Access to XML document

XML is not a programming language. We can't declare variables, set triggers events, loop 30 times, or do anything except using XML to describe information. XML does not have any built-in scripting languages. However, XML is the data structure of the fundamental part programming. So a language is needed to manipulate the data structures.

From the beginning, Java distinguished itself among programming languages by being platform-independent. Objects created in Java would run on any platform that supports Java. In a way, XML is platform-independent data. If the programmers accept the argument that programs and data go hand-in-hand, then they can see why Java and XML are such a perfect fit. So Java helps in its interaction with XML documents, but Java is not unique in that sense.

In order for a programming language to effectively interact with XML documents two types of transformations must be captured. First of all, the program must have a way to interact with the various nodes (elements, attributes, text, etc); second, it must have a way to navigate the hierarchical data structure of an XML document. So a parser is needed, which is a program that reads and interprets an XML document. It does the low-level interaction with a physical XML file or a stream of data representing an XML document; and exposes the elements within an XML document and hierarchical structure of the document. There are a number of parsers available in recent years. Such as:

- XML Parser from IBM
- Project X from Sun Microsystems
- DataChannel and Microsoft XML Parser
- XP by James Clark, etc.

When programmers know how to write XML document and DTD, and they have a parser in hand, for examples Project X, how are they going to work with them? Two tool sets can help: one lays on the document in time (SAX), the other in space (DOM). SAX and DOM are the main tools to process XML and DTD.

- SAX (Simple API for XML) [SUN–XML]

SAX associates an event with tag (opening or closing) and each block of text. It can be used to read an XML file sequentially. It's the event-driven, serial-access mechanism that doses element-by-element processing.

- DOM (Document Object Mode)

DOM [W3C_DOM] is a programming interface independent of a language and platform, which permits scripts to access and modify the content, structure and style for HTML and XML documents. The DOM Core defines the tag-document of XML as a tree-like structure of node objects; in other words it breaks up the document into a hierarchy. JAXP produced by SUN makes it possible to operate XML document by DOM and SAX.

- JAXP (Java API for XML Parsing)

JAXP lets developers easily build Java technology-based applications powered by XML for e-commerce, enterprise application integration and Web publishing. Specifically, it allows developers to read, manipulate, and generate XML documents using pure Java APIs. As an optional package, JAXP is not part of the JRE (Java Runtime Environment), but it can be added for specific development needs. JAXP is the new standard method for a Java platform-based application to plug in any XML-conformant parser.

JAXP is the parser, which reads XML text within an application and a standard that gives developers flexibility of choice in parsers. JAXP makes it easier for developers to get started with XML by providing a standard, seamless interface for applications to support any XML-conformant parsers. It fully conforms to XML standards established by the World Wide Web Consortium (W3C) and the XML community.
As shown in Figure 3.5, the JAXP APIs contained in the `jaxp.jar` file, is comprised of the `javax.xml.parsers` package. The package contains two vendor-neutral factory classes: SAXParserFactory and DocumentBuilderFactory that give programmers a SAX parser and a DocumentBuilder respectively. The DocumentBuilder, in turn, creates DOM-compliant Document object.

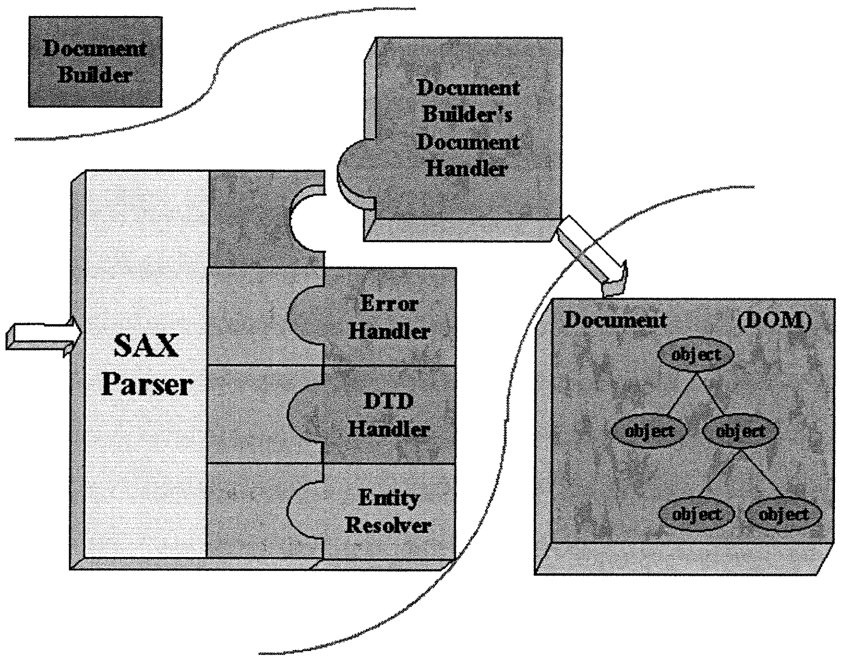


Figure 3.5 JAXP Parser Working Model [SUN-XML]

3.5 JSP technology and its XML solution

Java has traditionally been known as providing great Web client-side application support. Java is also a great platform for writing the server side Web-based applications. JSP (JavaServer Pages) [SUN_JSP] offers Web builders a very powerful way of dealing with unknown or thin client requirements. With the same features with Java, JSP is an excellent platform for writing client and server applications.

Using JSP technology, organizations are able to leverage existing Java platform expertise and create highly scalable enterprise applications. It has following features [JSP_XML]:

- Easy and Rapid Web Development, Deployment and Maintenance
- Emphasizing Reusable Components

- Separating Content Generation from Presentation
- Open Development and Widespread Industry Support
- Platform Independence
- Simplifying Page Development with Tags

JSP technology provides a number of capabilities that are ideally suited for working with XML. JSP pages can contain any type of text-based data, so it is straightforward to generate documents that contain XML markup. Also, JSP pages can use the full power of the Java platform to access programming language objects to parse and transform XML messages and documents. In particular, as part of the Java software environment, JSP pages can use objects that leverage the new Java APIs for processing XML data. Finally JSP technology provides an abstraction mechanism to encapsulate functionality for ease of use within a JSP page. Figure 3.6 shows the data accessing structure among JSP server, client and data source.

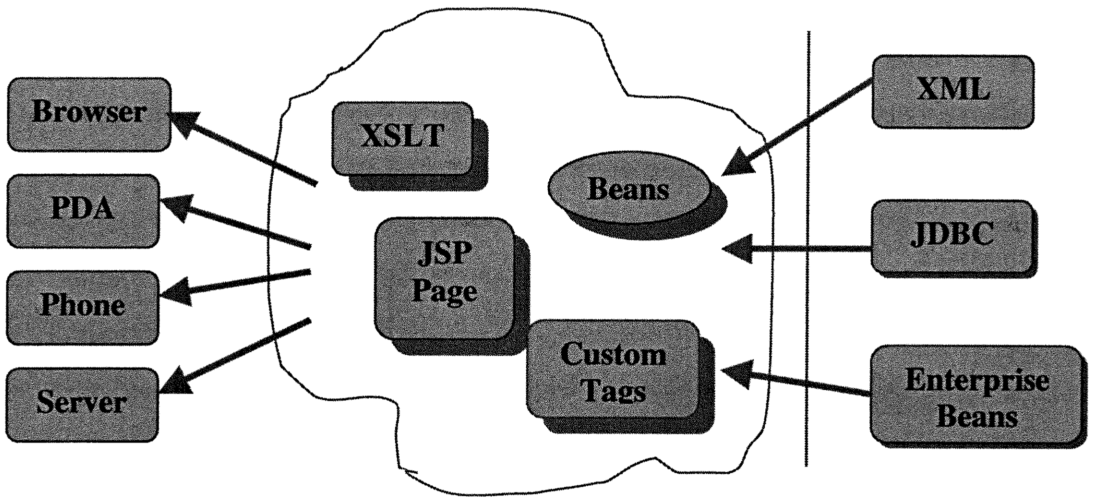


Figure 3.6 Data Accessing in JavaServer Pages Technology[SUN_JSP]

- Using multiple data sources

It is easy to use multiple data sources, including XML sources, in a JSP page. The data could be XML document, database, or object. A page may connect to a data source through a server-side object, transform the information into data abstractions, and then render the data using JSP elements.

- Converting XML to server-side objects and extracting object properties

XML/Java Binding technology (JSR 31) in the future will process XML document automatically by converting XML to server-side objects and extracting object properties, except for using DOM or SAX and encapsulating into a custom tag or a JavaBeans component in order to create these objects. An example is shown in Figure 3.7.

```
<%@ taglib uri="..." prefix="tl" %>
<html>
<tl:parse id="customer" type="Customer"
  xml="XML_Customer_URL"/>
<tl:parse id="saleBooks" type="BookInventory"
  xml="XML_Book_Inventory_URL"/>
<head>
<title>Welcome</title>
</head>
<body>
Welcome&nbsp;
<jsp:getProperty name="customer" property="lastName"/>
&nbsp;
<jsp:getProperty name="customer" property="firstName"/>
<table border="0" width="50%">
<tl:iterate id ="book" type="Book"
  collection="<%= saleBooks.getBooks() %>" >
<tr>
  <td>
    <jsp:getProperty name="book"
      property="title"/>
  </td>
  <td>
    <jsp:getProperty name="book"
      property="price"/>
  </td>
</tr>
</tl:iterate>
</table>
</body>
 </html>
```

Figure 3.7 An Example of Converting XML Data to Server-side Objects

- Generating Markup Languages Using JSP Pages

We saw a lot of different types of user-oriented clients for Web-based applications, such as PDAs, WAP-enabled mobile phones, and landline voice clients. JSP technology and XML are the natural partners for developing multilingual Web applications that use heterogeneous data sources and support multilingual clients.

As the world quickly moves toward "Write Once, Run Anywhere" capability-based solutions, JavaServer Pages will fast become a requisite technology for developing dynamic web pages [JSP features].

Chapter 4

Architecture of XMLFinder System

The description of a customer's requirement usually is more general and not exactly clear facing various products. So we have to utilize the ambiguous information and other information that the customer can provide, such as their preferences and constraints to help them find a satisfactory answer. Communication between servers or agents also poses a problem, where the data consumers and data providers all come together -- each talking a different language. The result is that the system looks like a "Tower of Babel" [CoCo, 1998]. Fortunately, XML -- the universal format data for data interchanging can tame this chaos by acting as the object model for distributed data.

4.1 Creation of XMLFinder agent system

We proposed a shopping agent system -- XMLFinder, which will give consumers the products' recommendation according to users' preferences and constraints. It would look like a product broker, which will find a list of products based on user criteria through a desirable user interface that prompts shoppers to input their constraints and gets information through standard search procedures.

Of course, there are some brokers' web sites that already exist on the Internet. The problem is that most agents need to do a lot of maintenance to translate data from different sources to a well-known format. As a result, in many instances browsers can't get data automatically. They have to first collect information and then assemble it in a useful format. A better way for vendors to publish their information would be through the use of a well-known data format. The XML standard meets most of the required criteria and provides a promising tool in the development of this critical area of Internet development. By taking advantage of existing features of XML, in combination with other new technologies, our multi-agent system (XMLFinder) would be unique as a web application in electronic commerce field for recommending products for e-shoppers.

XMLFinder is composed of two agents – an XMLAgent and a RecommendAgent. The first is responsible for the generation, integration and publishing of XML documents; the second is responsible for the retrieval of information by using agent techniques. These two agents work and interact with each other. Figure 4.1 shows the architecture of our system. It is constructed on three-tier architecture. Each tier has its own responsibilities; together they form a cohesive, flexible, and scalable shopping application, where XML will take an important role at each tier. The 'intelligent mechanism' would be introduced in next chapter.

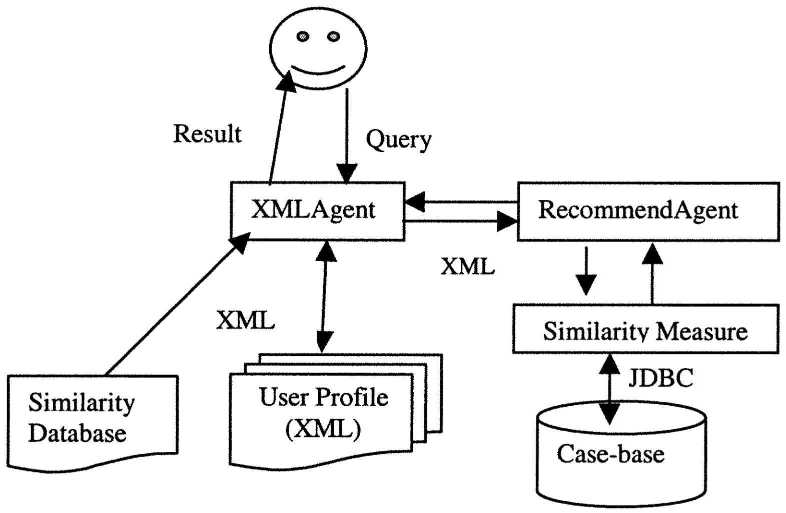


Figure 4.1 The Architecture of XMLFinder

4.2 Three-tier net architecture of XMLFinder

Why do we use the three-tier network architecture? The concept of tiers provides a convenient way to group different classes of architecture. Basically, if an application is running on a single computer, it has a one-tier architecture. If an application is running on two computers – for instance, a typical Web CGI application that runs on a Web browser (client) and a Web server – then it has two tiers. In a two-tier system, there are a client program and a server program. The main difference between these two is that the server responds to requests from many different clients, while the clients usually initiate the requests of information from a single server [Schussel, 2000].

Often, a two-tier application will need to store data on a server, usually, the information is stored on the file system; however, data integrity issues arise when multiple clients simultaneously ask server to perform tasks. That makes us adopt the three tier net structure.

A three-tier application adds a third program to the mix, usually a database, in which the server stores its data. It is an incremental improvement to the two-tier architecture. The flow of information is still essentially linear: a request comes from the client to the server; the server requests or stores data in the database; the database returns information to the server; the server returns information back to the client. The basic three-tier logic network architecture is shown in Figure 4.2.

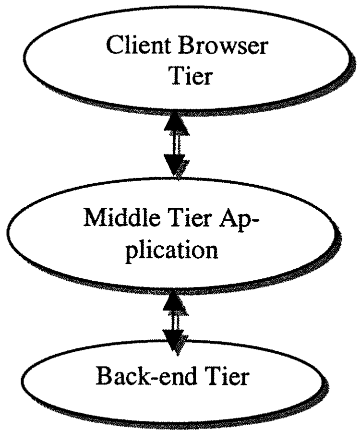


Figure 4.2 Basic Model of Three-tier Logic Network

Generally, our web-based shopping agent system (XMLFinder) follows this basic structure that consists of three logical tiers, in which each tier communicates with one another.

- A client tier, which consists of a Web browser,
- A middle tier, which has host's application, and
- A back-end tier, which hosts databases and data services.

The back-end tier is the third tier, which represents the XML document and relational database. The middle-tier is the business tier, which is the heart of the system. It represents a Java-based Web application. This tier corresponds with XML, either statically generated (served from an XML file) or dynamically generated (served by the JSP). Using XML instead of HTML, as we shall see, means that we can have dynamic content without sacrificing usability or interoperability. The first tier of our application includes a browser. It is also called the presentation tier; its main function is to send messages to the middle tier by encoding commands and arguments in HTTP requests.

4.3 Back-end tier

Back-end tier (also called data processing tier) is responsible for locating and storing data, in which data services will process an abstraction to the middle tier, so that the middle tier doesn't need to care about the location of data, or how to access, etc.

XML is used in the back end as an efficient storage format for any highly structured information -whether it is document or application data. If the data is hierarchical when it is in use, it makes sense to preserve the structure in storage. Storing it in another format requires the conversion of data and the inefficiency of imposing an inappropriate structure onto the data. In the case of a traditional relational database, richly structured data has to be decomposed into its elements, mapped to rows and columns, and then reconstructed when accessed. XML is a convenient and efficient storage format for any richly structured data.

XML is the ideal format for data processing tier as it is a common format for different data sources without special format and structure. In EDI processing, XML is taking the place of binary formats. Not only XML is easier to write than the proprietary EDI formats, but also XML can transmit HTTP traffic.

In our system, XMLAgent is responsible for part of data processing required for integrating XML documents from several data sources (such as ODBC database,) and transferring them to the middle tier. Figure 4.3 shows the back-end tier structure of our system. The back-end tier, which uses case representation, is discussed in the next chapter in detail. XML faces some problems in replacing the database despite being beneficial for data integrating and publishing [North, 2000]. Data volume can be a problem. Managing and searching XML data is hard when content exceeds a small number of files. Publishing projects deliver pages to a browser but they may need to search among a large amount of data in the database. Version control for collaboration and security of applications having sensitive information also pose a problem if there are multiple authors or users. Another issue is the need for sophisticated indexing and searching techniques. These are required to search a large collection of documents or to execute content-based queries. A content-based query can match text in an XML document, patterns in an image, a location in an area, or all three.

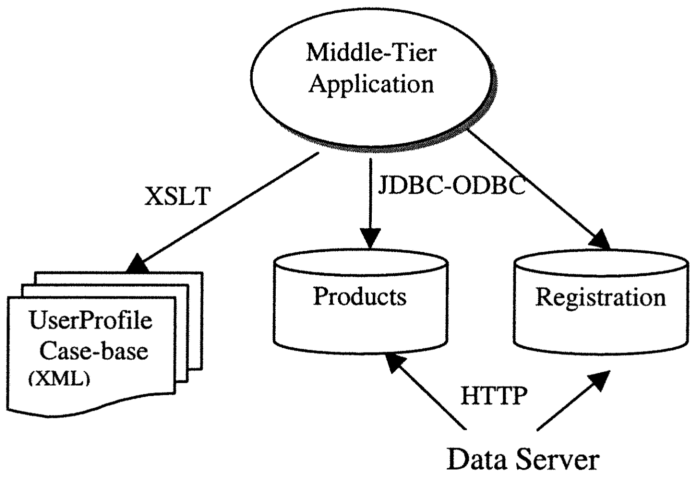


Figure 4.3 Back End Tier Model

We adopted a relational database (Microsoft Access) to store our products and users' registration information; therefore it's necessary for this system to exchange XML document and database data [XML-DB]. Fortunately, because of the widespread adoption of SQL, there are a lot of APIs for accessing SQL database. JDBC enables developers to use Java, SQL and advanced objects such as array and disconnected row-sets. We'll get well-formed file from database structure to access XML document by using JDBC-ODBC Bridge driver produced by Sun. Here are the examples showing the combination between XML document and database. Example 1 is a SQL structure that shows the creation of a product table of an Access database, which is constructed manually. Example 2 shows what the counterpart XML DTD might be for defining the product description as a case base in CBR.

Example 1: SQL for products

```
CREAT TABLE products (  PRONO           INTEGER,
                          BRAND           CHAR (50),
                          STYLE           CHAR (50),
                          PRICE           DOUBLE,
                          SIZE            CHAR (50),
                          COLOR           CHAR (50),
                          PICTURE CHAR (50)
                          primary key (PRONO) )
```

Examples2: DTD of case base for products

```
<xml version='1.0' encoding = 'us-assii' ?>
<!-- DTD for case base of users>

<!-- start of element declarations -- >

<!ELEMENT user (case)+ >
<ATTLIST user    uid #REQUIRED>

<!ELEMENT case (caseNo?, Pbrand, Pstyle, Psize, Pcolor, Pprice, date, time)>

<!ELEMENT Pbrand (value, weight)>
<!ELEMENT Pstyle (value, weight)>
<!ELEMENT Psize (value, weight)>
<!ELEMENT Pcolor (value, weight)>
<!ELEMENT Pprice (value, weight)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>

<!ELEMENT value (#PCDATA)>
<!ELEMENT weight (#PCDATA)>
```

4.4 Middle tier

Middle tier application, also known as business service tier, is responsible for exchanging data with client browser and interfacing to back-end tier data, aggregating information from necessary sources, processing it with internal business logic, and creating a set of information to be sent to the client.

In our system the middle tier application takes the role of RecommendAgent and XMLAgent. The former gets personalized content from client browsers, then runs on behalf of the user in order to perform a task; the later sets up an agent program by using the improved searching capability of XML using content tags. The agent can search from the information sources to get a combined data that matches the selection criteria from user. Figure 4.4 shows the main tasks of the middle tier in our system.

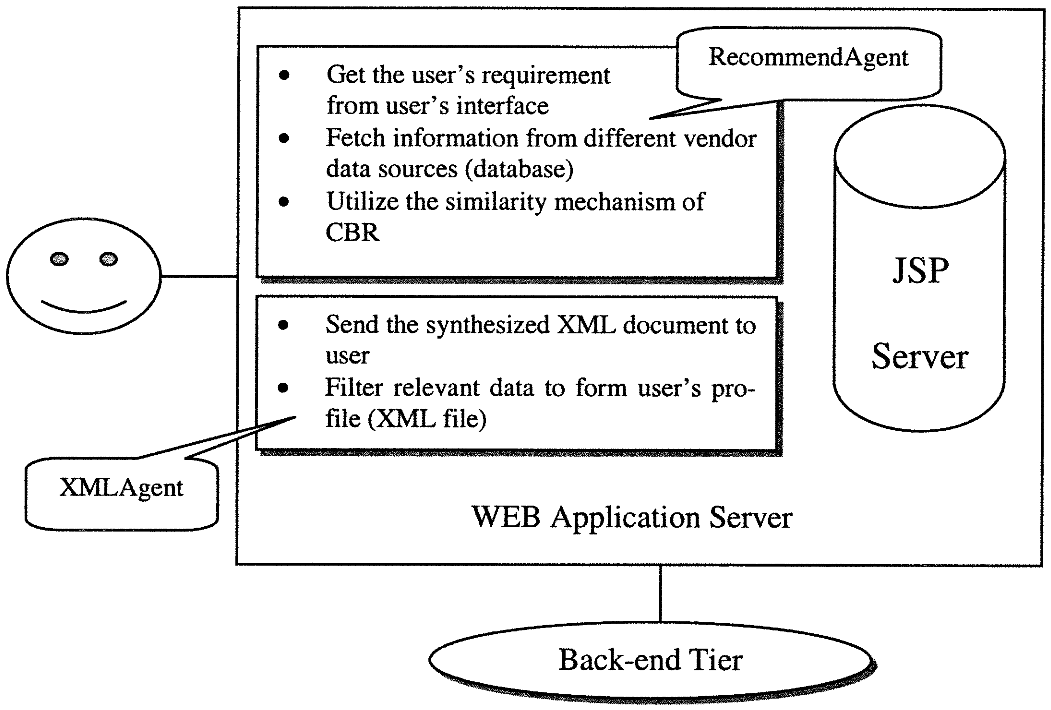

RecommendAgent
• Get the user's requirement from user's interface
• Fetch information from different vendor data sources (database)
• Utilize the similarity mechanism of CBR
JSP Server
• Send the synthesized XML document to user
• Filter relevant data to form user's profile (XML file)
XMLAgent
WEB Application Server
Back-end Tier

Figure 4.4 Middle Tier Application Server

The RecommendAgent is represented as a CBR mechanism, which uses a similarity engine to get mind-like searching results similar to the user's requirement. The intelligent RecommendAgent employs the user's preference and constraints to sift through the information provided by numerous content providers. The information is aggregated by the RecommendAgent and is represented as a profile form, which is then transferred to the client tier for publication.

The XMLAgent can be used to solve some of the most difficult issues related to application design and system integration. The middle tier is where communication and data sharing takes place. It includes transfers between application servers and communication, and data sharing between application servers and back-end storage engines. Because XML is self-describing, it can be used as a universal format for data interchange, which significantly simplifies the potential chaos in the middle tier.

4.5 Client tier

First-tier application, also called the user browser tier, or the client tier, serves as the interface between the end-user and an application running on a server. The client tier is the means by which end users interact with the application; it allows end users to view information generated by the application server and to input new information and submit request to the application server.

Our system is created on JSP platform, which supports several types of clients. JSP services can interact with their clients via dynamically generating HTML pages and forms. More sophisticated services will interact with their first-tier by directly exchanging business data. JSP and servlet are used to form this business data in a way that is easy for clients to work with. These clients can be both Java applets running on a web browser and Java technology-based programs.

XSL (XML Style Language) style sheet of an XML document are used to produce an output on the presentation tier. Because XSL can provide a clean separation between the content itself and the specific format required by a target application or output document. By separating content from the format in which it's used, XSL makes it easier to share information across multiple systems.

To apply an XSL style sheet of an XML document, a client must have an XSL processor. Currently IE5 [IE web site] is the only browser with an XSL processor. In the newer drafts, XSL provides not just a formatting capability, but also through XSLT, a full transformation capability for manipulating both tags and data content.

Examples3: XSL which translate a XML file

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<html>
<head></head>
<body>
  <table align="top">

    <form name = "form2" type ="post" action = "newcase.jsp" >
        <xsl:for-each select="ShopGuide/Entity">
        <tr>  <td width="8%"> <xsl:value-of select="CaseNo" />  </td>
                <td width="12%"> <xsl:value-of select="Brand" />  </td>
                <td width="12%"> <xsl:value-of select="Style" />  </td>
                <td width="12%"> <xsl:value-of select="Size" />  </td>
                <td width="12%"> <xsl:value-of select="Color" />  </td>
                <td width="8%"> <xsl:value-of select="Price" />  </td>
                <td width="8%"> <xsl:value-of select="Score" />  </td>
                <td width="8%">
                    <xsl:element name="input">
                      <xsl:attribute name="type" >checkbox</xsl:attribute>
                      <xsl:attribute name="name" >cart</xsl:attribute>
                      <xsl:attribute name="value" ><xsl:value-of select="ToCart" /></xsl:attribute>
                    </xsl:element></td>
                <td width="8%"> <a><xsl:attribute name="href">./image/<xsl:value-of   se-
                lect="Picture" /></xsl:attribute> show </a> </td>
                <td width="5%"> <xsl:value-of select="No" />  </td>
        </tr>
        </xsl:for-each>

    </form>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

- HTML Page Based Clients

A service can be presented directly to a user's web browser as dynamically generated HTML pages. JavaServer Pages technology is an easy way to dynamically compose these pages using a familiar scripting paradigm that combines HTML and Java technology-based code. In some cases, a service may require some fairly complex code. This can be handled by placing code in JavaBeans components and calling it from a JSP. A service can also be directly programmed in the Java programming language using a servlet, as shown in Figure 4.5.

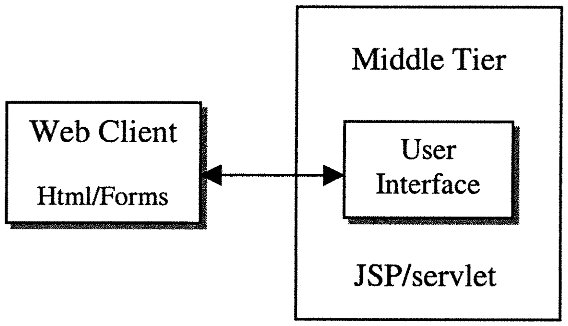


Figure 4.5 Presenting Services Directly to a Browser

- HTTP Content Based Clients

It is often useful to provide functionality directly at the client tier to help a user organize and interact with the service's information. In this case, the service exchanges raw content with the client instead of HTML pages. This content is typically in the form of XML documents that is exchanged between the client and the service using HTTP.

Typically this XML content is handled in the first-tier by JavaBeans components that are provided by the service in an applet that is automatically downloaded into a user's browser, as shown in Figure 4.6.

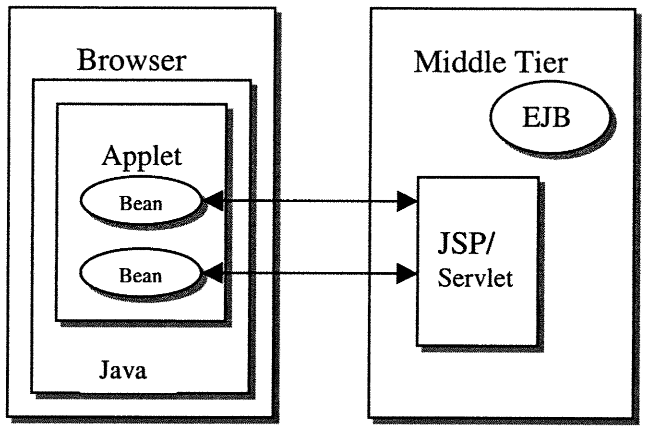


Figure 4.6 Providing JavaBeans Components to a Browser

In our system, the client tier serves the following purposes.

- User's registration
- Gathering information from the user
- Displaying recommendation information that has received from RecommendAgent
- Ending the transaction

In the following section, we provide an example to show how the client-tier works in our system.

4.6 Client-tier description of Fashion Agent

A prototype system of XMLFinder, *Fashion Agent* has been constructed. As a web application it gives recommendations when consumers buy clothes. For example, a lady comes to this agent and wants to buy a dress. The following steps are the procedures of the lady' dealing with this Fashion Agent from registering, querying, the agent's responding to and ending this transaction.

1. User's registration

Figure 4.7 shows the procedure of the user's registration. The procedure includes:

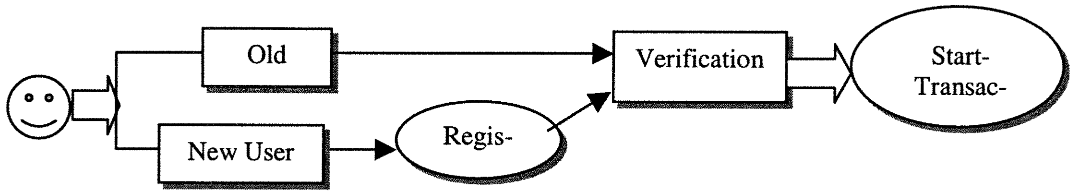


Figure 4.7 Processing of Registration

New user's registration (name, password, email, etc) to Fashion agent

- Old user sign-in procedure (name and password) to their agent system
- Browser's sending the message to the server for verification
- Server confirming the registration

2. Gathering information from the user, and then relaying back to the application server for further processing

Figure 4.8 shows the procedure of gathering information from users.

There are two pieces of information which are required from users, one is the preferences, which are given before each purchasing, another is the constraints, which show the likes or dislikes of users, which are stored in XML file form. The user can modify them at any time. The lady planning to buy her dress from the fashion agent might have the following preferences and constraints:

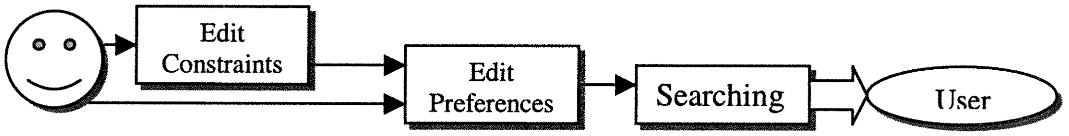


Figure 4.8 Gathering Requirement Information from Users

- Inputting preferences

As shown in Table 4.1, the lady will give the following attributes value pairs, in which the weights in the table will be discussed in chapter 5.

Table 4.1 User's Preferences Example

Attributes	Value	Weight
Brand	Polo	0.18
Style	Formal Dress	0.22
Color	Pink	0.2
Size	Medium	0.23
Price	50-100	0.17

- Inputting or editing user's constraints

Such as:

-- I never buy this brand : "Nike"

-- Don't recommend this style to me: "Short dress"

-- This color never fit me : "Black"

-- Don' t show me the products whose price are more than 5 times than my price.

3. Displaying information received from Fashion agent

- Display the recommended searching results according to the user's requirements in HTML format
- Display the user's Profile in XSL format

4. Ending the transaction

- User cancels her search processing
- User confirms the searching results and puts them in the shopping cart
- Fashion agent saves the confirmed results to case-base as the user's profile.
- User terminates the relationship with Fashion agent

Chapter 5

Methodologies and Algorithm

The basic idea of CBR is to solve new problems by comparing them to problems already solved [Aamodt &Plaza, 1994]. The key assumption is that if two problems are similar, then their solutions are probably also similar. In XMLFinder, a problem is typically the assignment of the particular product to a requirement stated by the customer. Products' retrieval based on k-Nearest Neighbor is used.

Nearest Neighbor (NN) is one of the oldest and more popular classifier techniques, which is wildly used in Case-Based Reasoning (CBR) system for similarity retrieval. In CBR cycle [Aamodt & Plaza, 1994] (a loop of retrieve, reuse, revise and retain steps) the retrieval phase is devoted to find the similar case among the cases stored in the case base depending on similarity criteria. This technique is at the base of many implementations of case-based reasoning systems [Aha & Goldstone, 1989] [Ricci & Avesani, 1995], because CBR systems have additional domain knowledge built into them compared with the traditional database system. The case base in CBR system is described as the training set in NN terminology. Therefore the researchers abstract the functions to assess the similarity of a given query to the cases in the case-base systems.

A similarity measure SIMILAR (q, c) can be used to represent the similarity between the query and the specific product attribute. Typically, similarity measure generates value 0 for query and specific case exhibiting no agreement among the selected attributes, value 1 when perfect agreement is detected and intermediate values for cases of partial agreement. To understand how such a similarity measure is used to find the best solution for a given problem, we can consider the cases being represented as a fixed-length vector n. These attributes may have several values and their values can be arranged to reflect some kinds of order. The problem space can now be seen as an n dimensional space (as shown in Figure 5.1). When a new query is represented to the system, the similarities to all the cases in the case base are calculated. The cases within a similarity range of a certain threshold are then presented to the user.

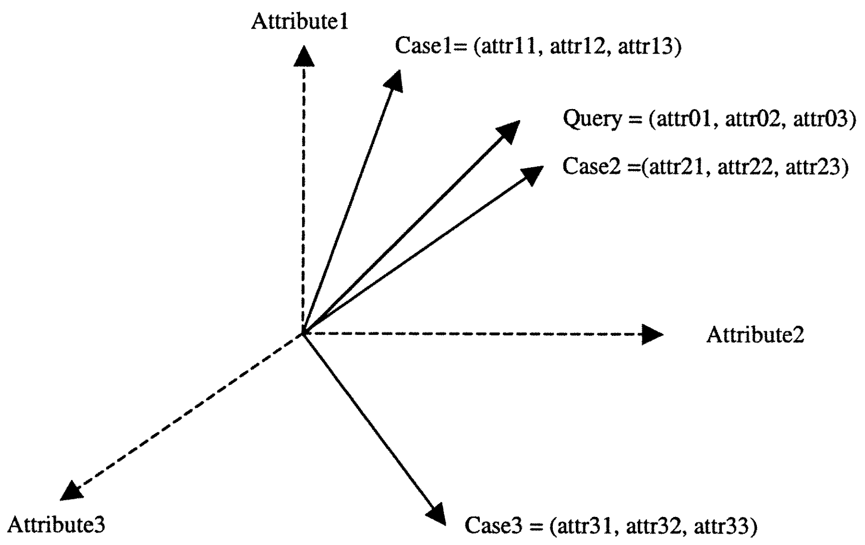

Attribute1
Case1= (attr11, attr12, attr13)
Query = (attr01, attr02, attr03)
Case2 =(attr21, attr22, attr23)
Attribute2
Attribute3
Case3 = (attr31, attr32, attr33)

Figure 5.1 Usage of similarity in CBR: cases are distributed over an n-dimension problem space

In the real world, the cases are described by several attributes, which may be represented by all possible pairs of attribute's values, or other data structure such as Boolean, taxonomy or a more complex object. In our prototype system, the cases are the descriptions of products -- clothes, which are described using several attributes (such as brand, style, color, etc). We adopt a metric in the context of nearest neighbor's method and find the weighted sum of local similarities. Then we show the first ten best cases to the users after scoring all the cases' similarities.

$$\mathrm{SIM}\,(q,c) = \sum_{i=1}^{n} \omega_i \ S_i(q_i\,, c_i)$$

Where SIM (q, c) is the global similarity between the query (q) and case (c), S_i is the local similarity between query and case at attribute i, n is the number of parameters of each case, ω_i is the weight given to parameter i, which shows the user's preferences to different attributes. Therefore, the heart of our agent system based on CBR is the computation of global similarity between a new case - the user's input (query) - and previous cases stored in a case base. The key is to compute: S_i (q_i , c_i) and ω_i initially in order to get the global similarities.

5.1 Local similarity measures

Cases are associated with qualitative and quantitative parameters called features or attributes. The CBR algorithm calculates the similarity between cases based on attribute-value pairs between the new and each historical case. We call the similarities as *local similarity.* The calculation of the local similarities depends on the type of the attribute and the range the attribute-value may take. The local similarities for discrete and continuous values are calculated in different ways.

- Discrete similarity

The similarity values of some attributes are listed in a table. They are the combination of the similarities to the possible attributes. There are two kinds of tables: symmetric and asymmetric table according to the native feature of the different attribute's values.

The asymmetric attribute values differentiate between a query attribute value of x and a case value of y, and vice versa (as shown in Table 5.1). For example, if the attribute is a "product's rating" and the user is looking for a product with three-stars, both a three-star and a five-star will satisfy this query (the similarity is 1). If the query is five-stars, on the other hand, the three-stars product does not satisfy the user's criterion (the similarity is 0.65).

Table 5.1 Asymmetric Similarity Attribute-value Pairs

Case / Query	One-star	Three-stars	Five-stars
One-star	1	1	1
Three-stars	0.6	1	1
Five-stars	0.4	0.65	1

The symmetric similarity attribute values are the same between a query attribute value of x and a case value of y, and vice versa (as shown in Table 5.2). For example, if the attribute is "color" and the user is looking for a "red" dress, only the "red" case can exactly satisfy the user's query (similarity is 1).

Table 5.2 Symmetric Similarity Attribute-value Pairs

Case / Query	Red	Rose	Blue
Red	1	0.85	0.5
Rose	0.85	1	0.45
Blue	0.5	0.45	1

- Continuous Similarity

The similarity values of some attributes such as the product price are continuous. It can't be listed in a table like discrete attributes. Therefore we need a function to describe it. In Fashion agent, we define a similarity function for the clothes' price as shown in Figures 5.2 and 5.3.

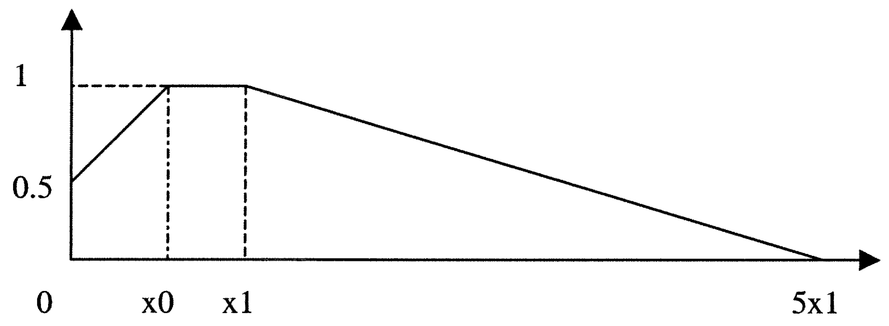


Figure 5.2 Continuous Similarity

$$\text{SIM}(q,c) = \begin{cases} 0.5+P/2X_0 & P< X_0 \\ 1 & X_0<P< X_1 \\ 1.25-P/4X_1 & X_1<P<5X_1 \\ 0 & P>5X_1 \end{cases}$$

Figure 5.3 ContinuousSimilarity Function

When users intend to buy products, they usually have a price range in mind. In most cases, this is not an exact price. For example, when a lady wants to buy a dress in the \$50–100 (x0–x1) range, she will be satisfied whether the price is \$60 or \$75 (the similarity is 1); however if the price is lower than her threshold (e.g. \$40), we define the similarity to be between 0.5 and 1. Although the cheaper price is preferred (defined as more than 0.5), it doesn't mean better. On the other hand, if the price is higher than the user's required limit, we define the similarity to be between 0 and 1. Also we define the similarity to be 0 if the price is more than 5 times the user's maximum price. In the previous example, if the lady's highest query is \$100, then the highest price she will consider paying is \$500. As a consequence, she will not consider any price higher than

$500, and the similarity is thus 0. The adaptation mechanism in our system adjusts the price function. We will discuss this at section 5.5.

5.2 Similarities storage and representation in XML

The Agent can get the continuous similarities immediately through the computation of functions. What we need is a way to get the discrete attributes' similarities. That means that we need an effective way to access the similarities in a table.

The values in the discrete similarity table have following features as shown in Table 5.3:

- Each attribute constructs a matrix, and
- Each matrix has different rows and columns, and
- Only half of data in the table is usable (limits at symmetric table in our system).

Table 5.3 Symmetric Similarities Stored in Separated Tables

Attribute A (Color):

Attribute: size	Red	Rose	Blue
1.Red	1		
2.Rose	A21	1	
3.Blue	A31	A32	1

Attribute B (Style):

Attribute: Style	Formal Dress	Casual Dress	Jacket	Short Dress
Formal Dress	1			
Casual Dress	B21	1		
Jacket	B31	B32	1	
Short Dress	B41	B42	B43	1

Relational database is very popular and mature to data storage in a long run. There would be two ways to store the discrete similarity data in a relational database. One way is to save each attribute in a different table, that means we have the tables with a number of attributes' amounts, but each table only has half of data to be used (as shown in Table 5.3). Another way is to put all the attributes' values in one table, in which each attribute occupies one row, that means the numbers of rows equal to the attributes' amounts, but the numbers of columns in each row are not equal. In other words, there is still much redundancy in this system (as shown in Table 5.4).

Table 5.4 All attributes' Similarities Stored in One Table

1	A21	A31	A32					
2	B21	B31	B32	B41	B42	B43		
3	……							

DBMS has a lot of advantages such as centralized control, data independence, and conflict resolution, etc. It's an effective database system for managing a large volume of data. However DBMS didn't show its benefits in our scenario due to factors such as high cost and relatively fixed data scheme. A point to remember is that the similarity structure is composed of the values of product attributes that are changeable.

How do we store the discrete similarity (e.g. symmetric similarity) values in a table? Adopting XML documents as the storage form of similarity data gives our system unique capabilities. This approach is more effective than saving it in a database or text file format since it is less costly than database storage (as shown in Tables 5.3 and 5.4). Moreover, it allows the extraction of the attribute's meanings directly from the XML tag. We refer to the XML document of similarity representation as the back-end similarities database (More detailed discussion is given on whether the XML document can be called a database, in [Ajit, 2000] [Quin, 2000]). Figure 5.4 shows an example of the similarities' representation of XML in our system.

```
<?xml version="1.0" encoding ="ISO-8859-1" ?>
<root>
      <attribute id="style">
        <ShortDress>
                <FormalDress>0.55</FormalDress>
                <CasualDress>0.75</CasualDress>
                <Jacket>0.45</Jacket>
                <Suit>0.65</Suit>
        </ShortDress>
        <FormalDress>
                <CasualDress>0.60</CasualDress>
                <Jacket>0.55</Jacket>
                <Suit>0.85</Suit>
        </FormalDress>
        <CasualDress>
                <Jacket>0.70</Jacket>
                <Suit>0.55</Suit>
        </CasualDress>
        <Jacket>
                <Suit>0.35</Suit>
        </Jacket>
      </attribute>
      <attribute id="size">
        <Small>
                <Medium>0.90</Medium>
                <Large>0.80</Large>
                <ExtraLarge>0.65</ExtraLarge>
        </Small>
        <Medium>
                <Large>0.80</Large>
                <ExtraLarge>0.65</ExtraLarge>
        </Medium>
        <Large>
                <ExtraLarge>0.80</ExtraLarge>
        </Large>
      </attribute>
      <attribute id="color">
        <Rose>
                <Sage>0.45</Sage>
                <Sand>0.60</Sand>
                <Wine>0.60</Wine>
        </Rose>
        <Sage>
                <Sand>0.90</Sand>
                <Wine>0.80</Wine>
        </Sage>
        <Sand>
                <Wine>0.80</Wine>
        </Sand>
      </attribute>
```

V. Methodologies and Algorithm

</root>

Figure 5.4 Examples of Similarities' Data Representation in XML

XML permits searching based on content tags, it is possible to search any number of information sources that matches the selection criteria, which have been indicated by the user. XML allows data to be stored in a format more suited to object-orientation, which offers a standard data format that allows objects to be created in a programming language like Java. XML exists in the form that can be transported across the Internet and processed. An XML document is a tree data structure (Table 5.5), as compared with a relational database, which is represented in a rectangular grid table (Figure 5.5). DOM and SAX parsers are effective to parse XML documents composed of node structures. XML nodes can be easily parsed into elements that store the data in object form and maintain the hierarchies of the nodes.

XML is also a convenient and efficient storage format for any richly structured data and a number of databases already support XML data formats. There is a growing industry support for a standard XML query language (XQL). XQL allows applications to search and retrieve information from a document in the same way as querying a database.

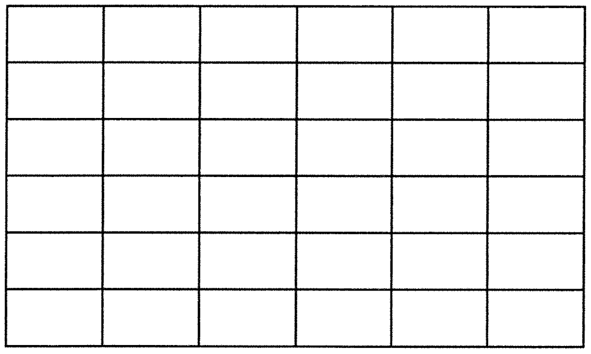


Table 5.5 Relational data model

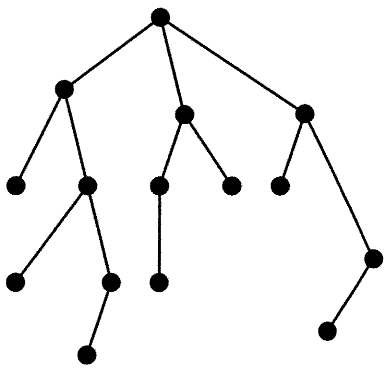


Figure 5.5 XML document model

Clearly, if data are stored in XML format on the back-end tier the job of the integration server is greatly simplified. The use of XML on the back end is being driven by the adoption of XML as a universal data format on the middle tier. We have tried this new area on our system.

5.3 Short-term profiling and weight modification

When shopping, except for his specific request, which is shown in attribute-value pairs, usually a client attaches a relative significance to the various parameters describing the product sought. For example, A lady who is looking for a dress may pay more attention to the brand than to the price. In this case, the agent should respect the user's preferences and pay more importance on the brand in its search. This kind of preference is temporary; hence it is called "short-term" [Aimeur & Vezeau, 2000]. If the user had a limited budget next time she does the shopping, the price would get the most important position as far as user preference is concerned and other attributes become secondary. In other words, we think the short-term profiling is a valid method used in CBR cycle. So we have this hypothesis that among the various critiques formulated by a client when searching for the ideal products, the short-term requirement is a good source of information to construct profile automatically. When a client frequently chooses his a preferable attribute such as the brand, he attaches greater importance to this concept. Therefore the shopping agent will keep on modifying the weights to grasp the user's preferences on different attributes.

Local similarity measurement functions re-rank feature relevance (weight) based on the feature values of the probe cases [Ricci & Avesani, 1999]. Therefore, it is possible to increase the feature relevance under certain conditions (user input) and decrease the relevance in other situations. In other words, the local weight may have to be changed depending on its task. A mathematical concept deficiency (Di), which measures what degree the system is considering this attribute to be susceptible to a user's critique, has to be defined as follows:

$$Di = \omega_i (1 - Si (qi, ci))$$

In this equation, ω_i represents the relative importance of attribute i. There are quantities to be modified when the user selects an attribute, which can be measured by a deficiency of the attribute. Once the deficiency of each parameter i is computed, we cover each parameter whose deficiency is higher than the preferable parameter, which we call D* .

$$\omega_i = \omega_i - (D_i - D_*) \qquad D_i > D_* \qquad \text{for all i}$$

Finally, the weight of the preferable parameter is more than at least one other attribute; i.e. the weight of the other attributes will be reduced.

$$\omega_* = \omega_* + \Sigma(D_i - D_*) \quad D_i > D_* \qquad \text{for all i}$$

Therefore, there will have been at least one parameter whose deficiency is higher than the critical deficiency.

As an example, Tables 5.6 and 5.7 represent the changed weights. When a user chooses a preferable attribute such as *brand*, we increase the weight of "brand" using the above algorithm.

Table 5.6 Before Computing New Weights

Attributes	Weight	Local Similarity	Deficiency
Size	0.20	0.75	0.05
Style	0.20	0.85	0.03
Color	0.20	0.80	0.04
Brand	0.20	0.90	**0.02**
Price	0.20	1	0.0

Table 5.7 After Computing the Weights

Attributes	Weight	Local Similarity	Deficiency
Size	**0.17**	0.75	0.05
Style	**0.19**	0.85	0.03
Color	**0.18**	0.80	0.04
Brand	**0.24**	0.90	0.02
Price	**0.2**	1	0.0

For example, a user is searching for a dress using the following search criteria:

Brand: Polo, Style: Formal Dress, Price: 50-100, Color: Pink, and Size: Small

The product database may have a dress in pink, which is the formal dress style with small size, Polo brand, and a price equal to $120. It is an almost perfect fit for her search criteria. However, because the price is $120, it doesn't fit into the price range exactly. The traditional technologies like SQL alone will not include this case in the result sets, because the case has to be matched exactly with the query. However through our case-base reasoning mechanism this case would present 99% match to the search criteria, the score of which case is 0.99 (computed by our metric). By utilizing the short-term profiling mechanism, we can grasp the short-term preferences to satisfy the user's requirement.

The initial weights can be equal to each of attributes if the user has not any preferences. For example, we initialize the new user's preferences to be equal (each initial weight is 0.2 if there are 5 attributes). However, old users can also save their preferences (including the weights) in their profile as a case (this will be discussed in the next section). At the time of next purchase, the new query's weights can be initialized by the weights of the user's last successful case (if it exists in the user profile), which reduces the optimized time and adaptation times.

5.4 Long-term profiling and case representation

The user's shopping habit such preferences or constrains usually lasts a longer time, it is called long-term profile. It gives us the important signal to grasp the user's requirement.

The long-term user profile is a collection of <attribute, value> pair. It is accessible and modifiable by user himself, and it includes two kinds of information. One is composed of the last successful purchase case, another is the user's constrains that the user can modify at any time. The availability of the user profile is important for our system, it presents a longer-term's case to join short-term profiling in the case-based reasoning implementation. Another unique feature of our system is that the case base of the user profile is constructed by XML file, which is possible to be exchanged with by other agents conveniently. Figure 5.6 shows the example of the constraints file in XML format, and we represent its DTD file in Figure 5.7.

```
<?xml version="1.0" encoding ="ISO-8859-1" ?>

<root>
  <user uid="maya">
    <case>
        <caseNo>1</caseNo>
        <Pbrand>
          <value>dELiAs</value>
          <weight>0.186</weight>
        </Pbrand>
        <Pstyle>
          <value>ShortDress</value>
          <weight>0.2</weight>
        </Pstyle>
        <Psize>
          <value>Large</value>
          <weight>0.186</weight>
        </Psize>
        <Pcolor>
          <value>Black</value>
          <weight>0.2</weight>
        </Pcolor>
        <Pprice>
          <value>46.0</value>
          <weight>0.342</weight>
```

```
            </Pprice>
            <date>2000-12-20</date>
            <time>9:25</time>
        </case>

        <case>
             <caseNo>4</caseNo>
           ... ...
        </case>
    </user>

    <user uid="maria">
        <case>
           ... ...
        </case>

    <user uid="david" />
</root>
```

Figure 5.6 An Example of User's Profile in XML

```
<xml version='1.0' encoding = 'us-assii' ?>
<!-- DTD for users'constraints -->

<!-- start of element declarations -- >

<!ELEMENT user (constraint)+ >
<ATTLIST user
             uid     #REQUIRED>

<!ELEMENT constraint (Cbrand, Cstyle, Ccolor, Cprice)>

<!ELEMENT Pbrand (#PCDATA)>
<!ELEMENT Pstyle  (#PCDATA)>
<!ELEMENT Pcolor (#PCDATA)>
<!ELEMENT Pprice (#PCDATA)>
```

Figure 5.7 An Example of User's Constraint Profile in XML

For example, a customer has following constraints:

C1: I never buy the 'Polo' brand

C2: Don't provide the 'short dress, to me

C2: I don't buy the 'black' dress

C2: The highest price should not be 6 times more than my query.

The name-value pair structure is used by XML document as a case representation means. This means that the communication between client and server is more robust. The application using a name-value structure will look for the name and then processes the value. The robustness results from the fact that XML doesn't require that elements appear in particular order. This method contrasts with using a specific record layout like relational database as the fields must be in specific order, otherwise error will occur, or data will be processed incorrectly.

5.5 Adaptation

Another point that makes CBR different from standard database approaches is its ability to not only retrieve existing cases, but also adapt their old solution to the newly presented problem and thus create new solutions that were not represented in the case base. In general, the researchers distinguish between three categories of products considering their ability to be customized:

- Unchangeable products, e.g., integrated circuits, books, etc.
- Products with few changeable features, e.g., vacations, houses, etc.
- Products with many changeable features, e.g., computers, cars, design tasks, etc.

Many products can't be adapted because they have no modifiable structure, such as books, which belong to the first category. Adaptation is indispensable if the system must perform some kind of configuration task, like interactive configuration of personal computers or cars. Adaptation is dependent on the application domain, using a process which can be more or less complicated. Computer domain is the most suitable domain in adaptation. In the computer shopping system, the transformational adaptation is adopted. Researchers employ a fixed set of transformation rules, which depend on the differences between problem attributes in the query and the problem attributes in the similar case for which the actual solution is modified. Transformational adaptation requires domain knowledge on how certain differences in the problem lead to differences in the solution.

One possible representation is a set of rules that perform certain actions given that the required preconditions for the actions are valid. For example, there is following representation in computer shopping system.

```
If (query.database and similar.database>5)
Do: soluton.diskspace = solution.diskspace + 2G

If (query.games >0 and similar.games=0)
Do: Object: joystick
```

In our fashion scenario, some products are configurable, such as dress and shoes. For example, some stores have the services of cutting the pants to the required size. We can use this as one of adaptation mechanisms. It is represented in this way:

```
if      query.pants = true and case.size=false
do:     case.size = true;
        case.price := case.price + $5;
```

To the author's knowledge, there is currently no commercial CBR-based product catalog that supports adaptation. The simplest kind of adaptation is null adaptation. where the case base is searched for a similar case containing the most similar user requirements and the solution from this case is taken to solve the current problem without any modification. Null adaptation means the adaptation is left to the user or it is even not necessary to adapt at all. This is the main adaptation mechanism we adopted in Fashion Agent.

Implementation of XMLFinder --- *Fashion Agent*

In order to illustrate the XMLFinder architecture and methodology, we constructed a *Fashion Agent* as a web application to give recommendations when consumers buy clothes. Because the next-generation agent-based system needs to run on a usable, interoperable and dynamic content requiring environment, we choose Apache's Tomcat 3.1 as our Web server with JSP engine to serve XML.

6.1 Selection of Web server technologies

There are many Web support technologies in the world. Apache and Microsoft-IIS are the most popular servers. Having compared Common Gateway Interface (CGI), Active Server Pages (ASP) and JavaServer Pages (JSP), we have selected JSP technology as our Web server foundation.

6.1.1 Web sever technologies -- CGI, ASP, or JSP

Common Gateway Interface, or CGI, has been the dominant interface for extending web server for years. Because CGI support was built into every web server on the market, CGI was a great choice for development tools and applications that could add dynamic capabilities to the web site.

CGI was developed at a time when most of the content on most web sites was static; The CGI script would then provide a limited amount of dynamic content on a fraction of the site. Traditionally, CGI languages include C, C++, and Perl, with Perl being the dominant CGI language. The main disadvantages of CGI are speed and the need for a separate program.

CGI scripts are still useful and will probably continue to be useful for years to come, however, they are not suitable for the construction of a large production site, especially for the e-commerce application.

Other major approaches to dynamic content in use today are based on Server side scripting of ASP and JSP. Microsoft Active Server Pages (ASP) and JavaServer Pages (JSP) have many similarities. Both enable developers to separate programming logic from page design through the use of components that are called from the page itself. Also, they both provide an alternative to creating CGI scripts that makes page development and deployment easier and faster. The biggest difference between JSP and ASP technologies lies in the approach to the software design itself.

Because it uses ActiveX controls for its components, ASP technology is basically restricted to Microsoft Windows-based platforms. Offered primarily as a feature of Microsoft IIS, ASP technology does not work easily on a broader range of Web servers because ActiveX objects are platform specific.

However, JSP technology adheres to the Write Once, Run Anywhere philosophy of the Java architecture, It is modeled after ASP, and borrows many of its features: separating content from design and built-in objects like sessions, application variables and server-side-includes. JSP uses the powerful Java programming language, which gives developers more programming options than the scripting languages with ASP. JSP has the advantage of cross-platform support with its generic Java byte codes that run on any Java compliant machine.

ASP runs on other platforms as well but requires that COM objects either be written in Java or be compiled on the same platform that the server runs on. JavaBean components are easier to develop than COM objects and may be distributed across several different servers without needing to be re-compiled.

While both ASP and JSP use a combination of tags and scripting to create dynamic Web pages, JSP technology enables developers to extend the JSP tags available. JSP developers can create custom tag libraries, so page authors can access more functionality using XML-like tags and depend less on scripting. With custom tags, developers can shield page authors from the complexities of page creation logic and extend key functions to a broader range of authors. Table 6.1 shows the main differences between ASP and JSP.

Table 6.1 Comparisons between ASP and JSP

	ASP technology	**JSP technology**
Web Server	Microsoft IIS or Personal Web Server	Any Web server, including Apache, Netscape, and IIS
Platforms	Microsoft Windows	Most popular platforms, including the Solaris Operating Environment, Microsoft Windows, Mac OS, Linux, and other UNIX platform implementations
Reusable, Cross-Platform Components	No	JavaBeans, Enterprise JavaBeans, custom JSP tags
Security Against System Crashes	No	Yes
Memory Leak Protection	No	Yes
Scripting Language	VBScript, JScript	Java
Customizable Tags	No	Yes

The power of JSP is not limited to big enterprise or companies that can afford an application server, at the 1999 JavaOne conference, Sun announced a partnership with the makers of the Apache web server to provide full support for JSP under Apache. The ***Jakarta*** project – ***Tomcat*** [Apache.org], as it has been named, will allow anyone with a computer to develop and deploy JSPs (free download from [Apache.org]). Therefore Tomcat -- Jakarta Project is chosen as JSP engine for our Fashion Agent system.

6.1.2 The Jakarta Project -- Tomcat

Sun Microsystems has delivered the latest versions of JavaServer Pages^{TM} (JSP) and Servlets source code to the Apache Software Foundation to be developed and released under the Apache development process as the official JSP 1.1/Servlets 2.2 reference implementation. Apache, Sun, and a variety of other companies and individuals are openly developing a robust reference implementation that is freely available to any company or individual. This reference implementation developed under the project name Jakarta and code-named Tomcat, will be the only reference implementation available. This implementation is available to any company or developer to be used in Web servers, development tools, and to create dynamic, interactive Web sites.

The mission of the Jakarta Project is to provide commercial-quality server solutions based on the Java Platform that is developed in an open and cooperative fashion. The flagship product, Tomcat, is a reference implementation of the Java Servlet and JavaServer Pages^{TM} (JSP) Specifications, which can run standalone as well as integrated into the Apache Web Server. This reference implementation provides an operational definition for the Enterprise Java^{TM} JSP and Servlet application programming interfaces (APIs). All final releases of the Servlet and JSP implementations will conform to the latest versions of the Servlet and JavaServer Pages specifications available from Sun. These specifications are defined and evolved through the Java Community Process and can be found at [SUN_Servlet] and [SUN_JSP].

6.1.3 Configuration to Tomcat 3.1

Our system runs on Windows NT, in which JSP server -- Tomcat is setup and configured in the following steps.

- Download and install the Tomcat binaries from the Jakarta Project at the Apache Software Foundation.
- Set the `JAVA_HOME` environment variable to the top-level JSDK directory For windows, the command would be the following, depending upon the JSDK version and drive:

  ```
  SET JAVA_HOME=C:\jdk1.2.2
  ```
- Start the web server. This involves running the startserver script from the `jakarta-tomcat\bin` directory.
- Create an HTML file with JSP content. In the `jakarta-tomcat\webapps\ROOT` directory, create `.jsp` file.

6.1.4 Running environment of Fashion Agent

We need the following running environment for Fashion Agent.
First, We select a Web server with JSP engine named Apache's Tomcat 3.1;
Second, we need a Web browser that supports XSL Translations (XSLT) that currently leaves us with one choice: Microsoft Internet Explorer 5, Netscape (even the Netscape 6 preview release) will not work right now;
Third, we need Java JDK. We use JDK1.3. If there are applets, a plug-in will be needed.
Fourth, the code requires Sun's Project X -- XML parser [SUN_XML]. (Any W3C-component DOM parser should work).

6.2 The domain

To solve the information overload, as we mentioned, the shopping agents recommend products from books, CDs, to software, hardware, almost everything. The domain we are considering contains direct support to recognized regular customers. There is an offering of products for which the customers apply their preferences and constrains. This provides a base for the long-term profiling mechanism.

The clothes domain was chosen as the context for this thesis because it is a common product with common characteristics (such as brand, color, etc.). There is a potential to provide an attractive web site, which most people can relate to. Also purchasing desired clothes need much time and energy than some products, hence the need for the clothes' agent is needed. This choice was also made simply because the clothes domain is familiar to the author.

For the purpose of the modeling, the domain is required to have some degree *of object-persistence*. With object-persistence we mean that the interest for a certain type object is likely to be persistent over time. The 'clothes' domain has object-persistence in the sense that the interests for a brand, for example, last over time, whereas the persistence of a product such as a newspaper is much lower. Today's news is less relevant tomorrow. However, clothes domain is more persistent. For example, the style, color or brand maybe not change in the same way as a person's characteristic or occupation does.

Our case base includes 200 cases for clothes that are described by the following five attributes: brand, style, color, size, and price range. Those cases are made up of 13 brands, 6 kinds of style, 10 kinds of color, 5 kinds of size, and 7 kinds of price range (as shown in Table 6.2) and more than 200 pictures of those clothes. Each attribute's local similarities are defined between 0 and 1 and are represented in an XML file.

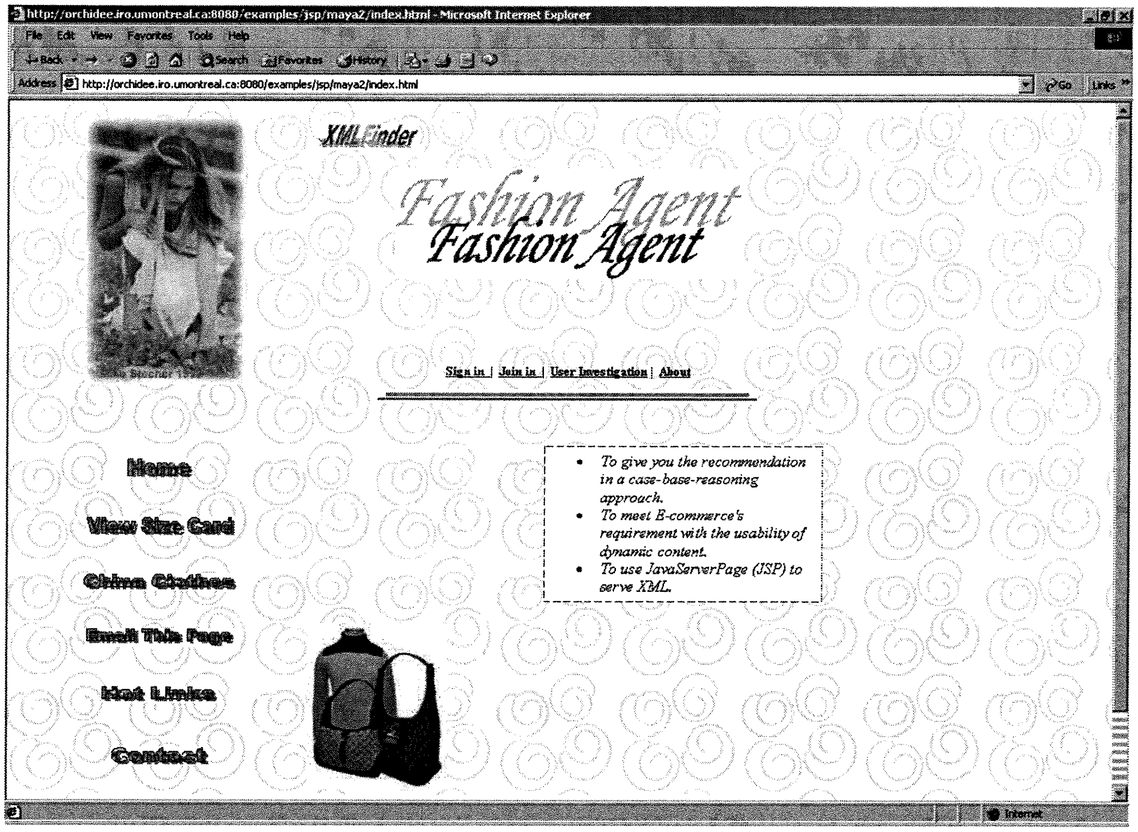
Table 6.2 Attributes and Values in Clothes Domain

Brand	**Style**	**Color**	**Price Range**	**Size**
BestProm Dress	Short Dress	Black	\$0–\$30	Extra Small
Bisou Bisou	Jacket	Blue	\$30–\$50	Small
Chadwick's	Suit	Green	\$50–\$100	Medium
Chelsea Nites	Casual Dress	Pink	\$100–\$300	Large
dELiAs	Formal Dress	Purple	\$300–\$500	Extra Large
Gap	Long Dress	Red	\$500–\$700	
Gordmans		Rose	\$700–\$1000	
Isabella Bird		Sage		

Laura		Sand		
Molly Malloy		White		
Nordstorm		Wine		
Polo				
Spiegel				

6.3 Implementation of Fashion Agent

As a complete shopping agent, we have constructed a web application which includes a lot of information about fashion, such as the level of interest in Chinese clothes, the clothes size recommendation table, pictures of various kinds of clothes and fashion models and hot links to various clothes stores, as shown in Figure 6.1. However, in this thesis we just focus on describing the recommended mechanism in XMLFinder system.

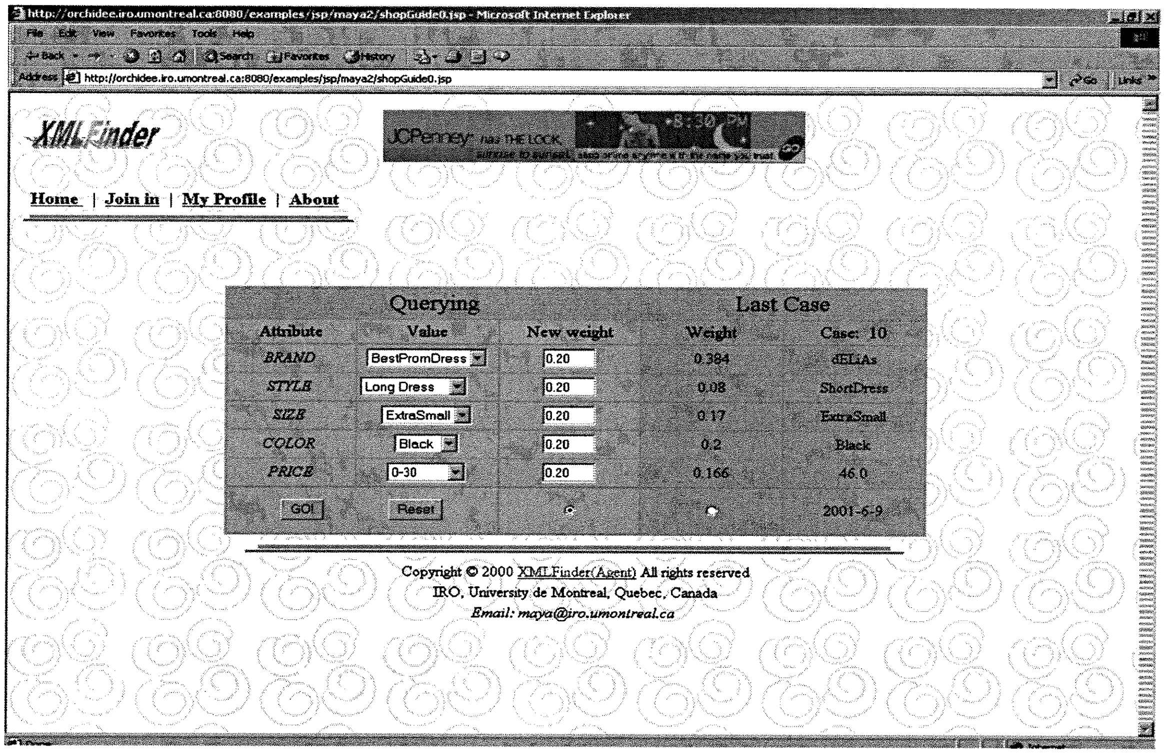

http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/index.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/index.html
Go Links
XMLFinder
Fashion Agent
Sign in | Join in | User Investigation | About
Home
View Size Card
China Clothes
Email This Page
Hot Links
Contact
Internet

- *To give you the recommendation in a case-base-reasoning approach.*
- *To meet E-commerce's requirement with the usability of dynamic content.*
- *To use JavaServerPage (JSP) to serve XML.*

Figure 6.1 First Page of Fashion Agent

6.3.1 Starting up the *Fashion Agent*

The registered user can start using the agent after log in. The agent page is opened (see Figure 6.2) after registering successfully. In this page, the user will provide the preferences and weights when she starts her agent for the first time. Each attribute has a corresponding weight. It can be set to the initial values or the weight of the last successful case. Users are not allowed to modify the weights freely (except for setting their initial values). This is a technical task allocated to the agent. After providing the preferences, the user just clicks ***GO***! To let the agent do the search.


http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/shopGuide0.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/shopGuide0.jsp
Go Links
XMLFinder
JCPenney has THE LOOK
Home | Join in | My Profile | About

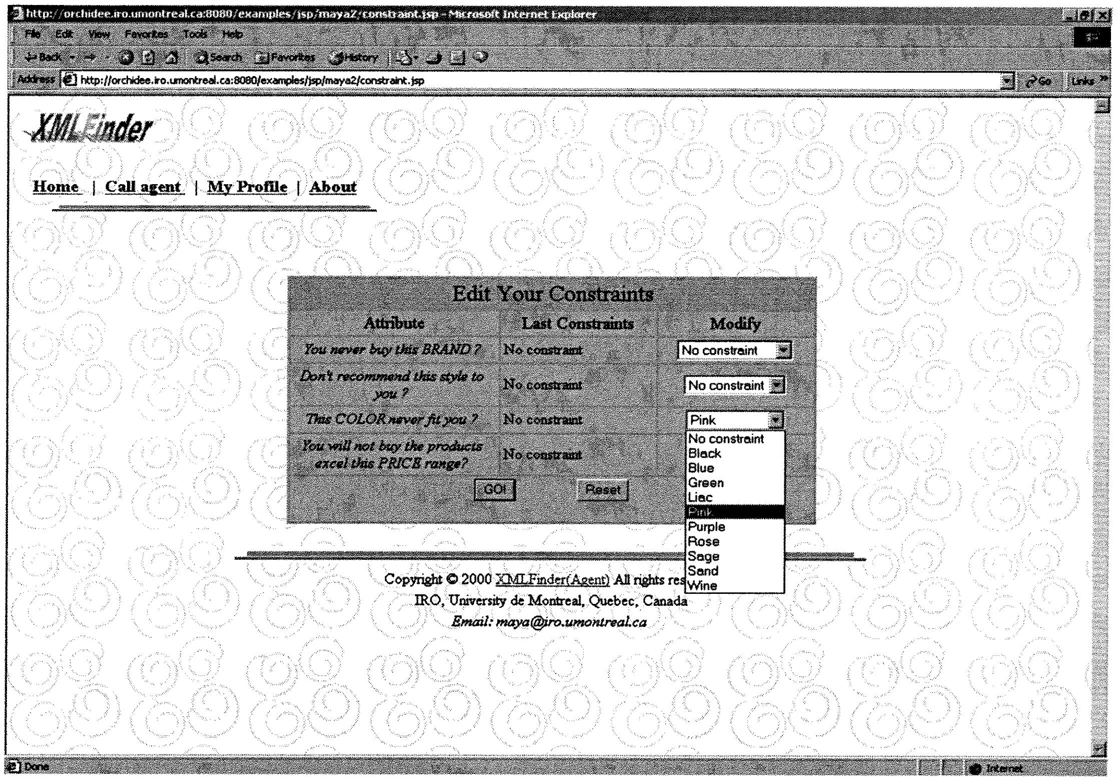
Querying			Last Case	
Attribute	Value	New weight	Weight	Case: 10
BRAND	BestPromDress	0.20	0.384	dELiAs
STYLE	Long Dress	0.20	0.08	ShortDress
SIZE	ExtraSmall	0.20	0.17	ExtraSmall
COLOR	Black	0.20	0.2	Black
PRICE	0-30	0.20	0.166	46.0
GO!	Reset	○	○	2001-6-9

Copyright © 2000 XMLFinder(Agent) All rights reserved
IRO, University de Montreal, Quebec, Canada
Email: maya@iro.umontreal.ca

Figure 6.2 Querying Page

6.3.2 Editing the constraints

The user can give his constraints anytime after registry, as mentioned in section 5.4. The agent will filter out the cases according to the user's constraints. The price range is adapted to the price function set by the user. Higher price range enables the user to have a wider selection range with less change in similarity value (the detail of the adaptation algorithm is represented in section 5.5). If there is no constraint provided by the user on price attribute, the range value defaults to 5. Figure 6.3 shows the user's constraints editing page.


http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/constraint.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/constraint.jsp
Go Links
XMLFinder
Home | Call agent | My Profile | About
Done
Internet

Edit Your Constraints

Attribute	Last Constraints	Modify
You never buy this BRAND ?	No constraint	No constraint
Don't recommend this style to you ?	No constraint	No constraint
This COLOR never fit you ?	No constraint	Pink
You will not buy the products excel this PRICE range?	No constraint	

No constraint
Black
Blue
Green
Liac
Pink
Purple
Rose
Sage
Sand
Wine
GO!
Reset

Copyright © 2000 XMLFinder(Agent) All rights res
IRO, University de Montreal, Quebec, Canada
Email: maya@iro.umontreal.ca

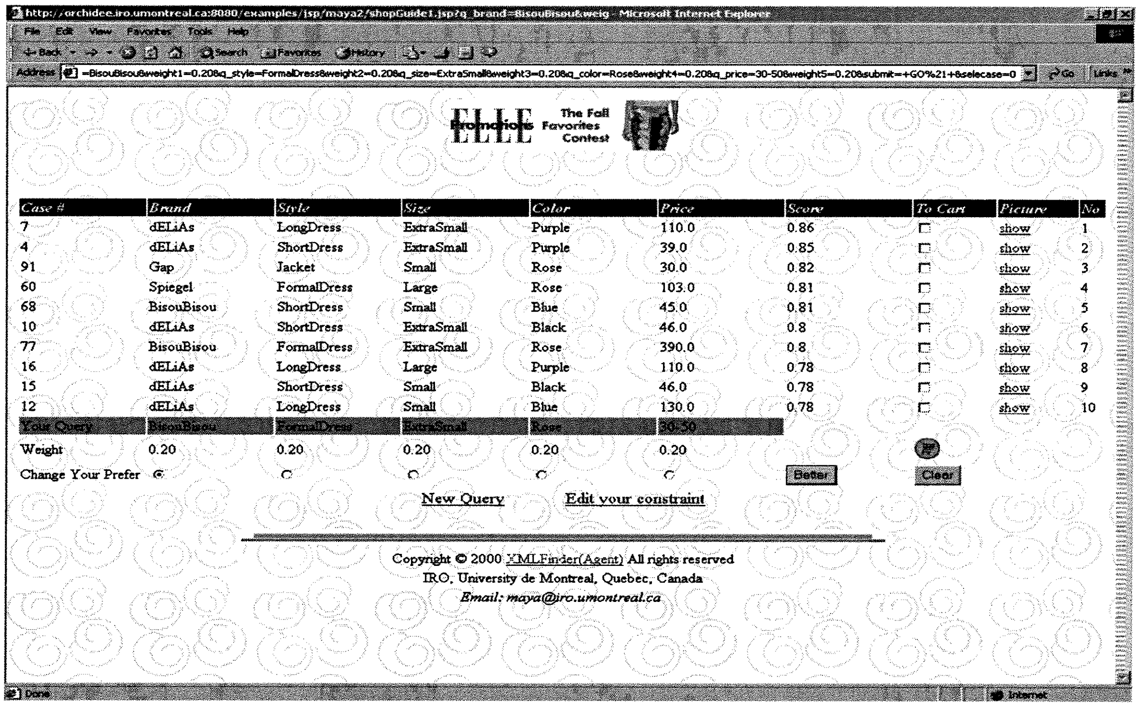
Figure 6.3 Editing Constraints

6.3.3 Optimizing the recommendation results

The agent will provide the first ten recommended results through scoring each case according to the preferences and constrains provided by the user as shown in Figure 6.4.

If the user is not satisfied with the results and still keeps the same query, she can click a button called *Better* to change the weight option. This button is a means of improving the effect of an attribute. The agent will score each case again. The further recommended results are shown in Figure 6.5, which are scored according to the user's further requirement by clicking "better" on brand.

By clicking the button called *Show picture*, the user can view pictures of clothes and the models of each case in order to make a more informed decision.


http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/shopGuide1.jsp?q_brand=BisouBisou&weig - Microsoft Internet Explorer

Address: =BisouBisou&weight1=0.20&q_style=FormalDress&weight2=0.20&q_size=ExtraSmall&weight3=0.20&q_color=Rose&weight4=0.20&q_price=30-50&weight5=0.20&submit=+GO%21+&selecase=0

Case #	Brand	Style	Size	Color	Price	Score	To Cart	Picture	No
7	dELiAs	LongDress	ExtraSmall	Purple	110.0	0.86	☐	show	1
4	dELiAs	ShortDress	ExtraSmall	Purple	39.0	0.85	☐	show	2
91	Gap	Jacket	Small	Rose	30.0	0.82	☐	show	3
60	Spiegel	FormalDress	Large	Rose	103.0	0.81	☐	show	4
68	BisouBisou	ShortDress	Small	Blue	45.0	0.81	☐	show	5
10	dELiAs	ShortDress	ExtraSmall	Black	46.0	0.8	☐	show	6
77	BisouBisou	FormalDress	ExtraSmall	Rose	390.0	0.8	☐	show	7
16	dELiAs	LongDress	Large	Purple	110.0	0.78	☐	show	8
15	dELiAs	ShortDress	Small	Black	46.0	0.78	☐	show	9
12	dELiAs	LongDress	Small	Blue	130.0	0.78	☐	show	10
Your Query	BisouBisou	FormalDress	ExtraSmall	Rose	30-50				
Weight	0.20	0.20	0.20	0.20	0.20				
Change Your Prefer	●	○	○	○	○	Better	Clear		

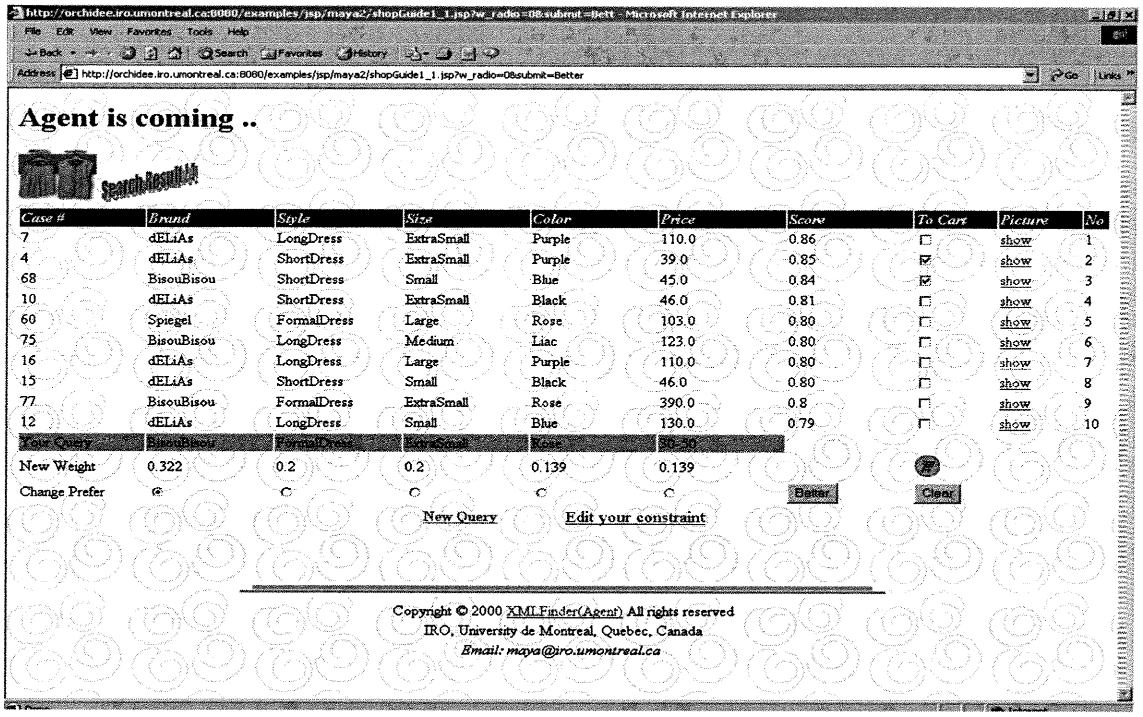
New Query   Edit your constraint

Copyright © 2000 XMLFinder(Agent) All rights reserved
IRO, University de Montreal, Quebec, Canada
Email: maya@iro.umontreal.ca

Figure 6.4 Recommended Results

6.3.4 Update the user's profile

The agent almost finishes his task in our prototype system. The last step is letting the user choose one case as the next purchase's reference. This will be saved in the user's profile. Figure 6.6 shows the shopping cart page.



http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/shopGuide1_1.jsp?w_radio=0&submit=Better

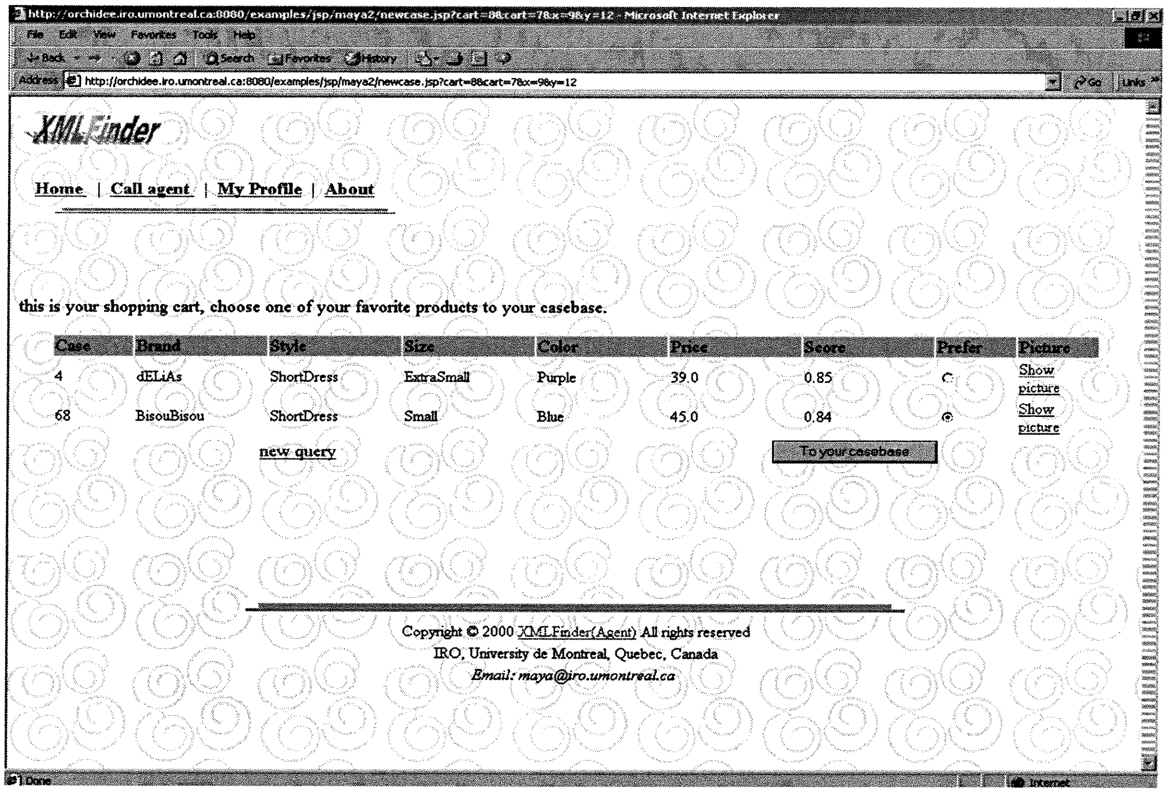
Agent is coming ..

Case #	Brand	Style	Size	Color	Price	Score	To Cart	Picture	No
7	dELiAs	LongDress	ExtraSmall	Purple	110.0	0.86	☐	show	1
4	dELiAs	ShortDress	ExtraSmall	Purple	39.0	0.85	☑	show	2
68	BisouBisou	ShortDress	Small	Blue	45.0	0.84	☑	show	3
10	dELiAs	ShortDress	ExtraSmall	Black	46.0	0.81	☐	show	4
60	Spiegel	FormalDress	Large	Rose	103.0	0.80	☐	show	5
75	BisouBisou	LongDress	Medium	Liac	123.0	0.80	☐	show	6
16	dELiAs	LongDress	Large	Purple	110.0	0.80	☐	show	7
15	dELiAs	ShortDress	Small	Black	46.0	0.80	☐	show	8
77	BisouBisou	FormalDress	ExtraSmall	Rose	390.0	0.8	☐	show	9
12	dELiAs	LongDress	Small	Blue	130.0	0.79	☐	show	10
Your Query	BisouBisou	FormalDress	ExtraSmall	Rose	30-50				
New Weight	0.322	0.2	0.2	0.139	0.139				
Change Prefer	●	○	○	○	○	Better	Clear		

New Query   Edit your constraint

Copyright © 2000 XMLFinder(Agent) All rights reserved
IRO, University de Montreal, Quebec, Canada
Email: maya@iro.umontreal.ca

Figure 6.5 Optimized Recommended results



http://orchidee.iro.umontreal.ca:8080/examples/jsp/maya2/newcase.jsp?cart=8&cart=7&x=9&y=12

XMLFinder

Home | Call agent | My Profile | About

this is your shopping cart, choose one of your favorite products to your casebase.

Case	Brand	Style	Size	Color	Price	Score	Prefer	Picture
4	dELiAs	ShortDress	ExtraSmall	Purple	39.0	0.85	○	Show picture
68	BisouBisou	ShortDress	Small	Blue	45.0	0.84	●	Show picture

new query   To your casebase

Copyright © 2000 XMLFinder(Agent) All rights reserved
IRO, University de Montreal, Quebec, Canada
Email: maya@iro.umontreal.ca

Figure 6.6 Shopping Cart

6.4 Discussion on generalization

XMLFinder is based on a general architecture, which can be applied in many domains, such as “selling books, CD”, “travel package”, “selling used cars”, “renting apartments”, etc. The approach we proposed in XMLFinder can directly be used in some domains, such as clothes, travel package, etc. However, because the approach is based on the attributes of ‘products’, there are some differences when it is applied in other domains. The “renting apartment” domain is used as an example in the following section to show the differences with the XMLFinder as used in the clothes domain.

- Similarity measurement

The main attributes in renting apartment domain are: living area, numbers of bedroom, accommodation type, rental price, required date (e.g. right now, 1 month, 2 month,...), and parking space. The similarities’ representation of these attributes all fall in the three categories: symmetric similarity (living area, numbers of bedroom, parking space), asymmetric similarity (accommodation type), and continuous similarity (rental price, required date), some of which are similar to the clothes domain. However, the similarity functions of the attributes “rental price” and “required date” may be different e.g. the price of clothes and renting apartment tend to have different values.

- Short-term profiling

XMLFinder has a hypothesis: the customers have no preferences when they initially use the agent system, which means the initialized weights of all attributes are equal. We find there is no difference between clothes domain (5 attributes) and renting apartment domain (6 attributes). However, in books domain, there seems to be an important attribute namely the book’s “title” which readers pay more attention to. So we have to give a higher weight to "title" attribute than others at any time.

- Long-term profiling

People have different reasons when they change their apartments. So the current preferences may be not useful for the future situation. This suggests that it is not necessary to create the long-term profiles of the customers in renting apartment domain unlike the book and clothes domains.

- Adaptation

Adaptation is domain-independent. Different adaptation mechanisms may be used in different domains. In renting apartment and book domain, because there are no configured parts to be adapted, the adaptation is not needed. However, in used car domain, the cars' parts are often assembled and changed, so the adaptation mechanism may play a key role in the case of used cars.

Chapter 7

Experiment and Evaluation

In order to test the validity of the approach we proposed in XMLFinder, an experiment was carried out where a group of testers were asked to run the system and give their feedback. This chapter reports on the results of this experiment. The objectives of this experiment are as follows:

- To find the performance of short-term profiling through interacting with the system by criticizing and modifying particular attributes.
- To find the performance of the long-term profiling by comparing the search results both in the presence and absence of constraints.
- To find the performance of combining the short-term and the long-term profiling.

7.1 Testing conditions and methodology

The method of experiment and evaluation is based on the collection of the user's feed-back. The respondents are asked to fill in a testing form when they run the agent system. In order to analyze the performance of Fashion Agent clearly, the whole recommended mechanism is tested in eight separate testing conditions C1-C8 (as shown in Figure 7.1). In the experiment, the respondents are instructed to go through the following procedure.

1. Users run the system.

The users provide their query first, and then the agent gives the products' recommenda-tion under the eight testing conditions (C1-C8) respectively after they register to this system.

2. Users evaluate the recommended results provided by Fashion Agent.

They are asked to reorder the recommended results provided by Fashion Agent depending on their own requirements, and then the reorder results are filled in the testing form (Table 7.1). The range of reordering is from 1 to 10 (1= most satisfied). For example, a tester has following order -- 2, 1, 9, 3, 7, 4, 5, 6, 10, 8 in the first testing condition C1. He fills them in the first line in Table 7.1, then keeps testing and reordering in C2~C8.

C1: Without constraints in the long-term profiling, and no preference in short-term profiling (simple searching)

C2: Without constraints in the long-term profiling, select one preferable attribute X

C3: After testing condition 2, select attribute X again

C4: After testing condition 3, select another preferable attribute

C5: With constraints in the long-term profiling, and no preference in short-term profiling

C6: With constraints in the long-term profiling, select one preferable attribute X

C7: After testing condition 6, select attribute X again

C8: After testing condition 7, select another attribute.

Note: *In this evaluation, the preference in short-term profiling is based on the temporary preferences to a certain attribute; the long-term profiling here means the constraints of the values of one or more attributes are provided by users. The more details about the short-term and long-term profiling are discussed in chapter 5.*

Figure 7.1 Testing Conditions in the Experiment

3. Users rate the agent system.

They are asked to rate the recommended results under eight testing conditions in a linear seven-point scale, ranging from 1 to 7, where 1=not satisfied, 4=average, and 7=excellent results. The satisfied degree can be considered very successful on these features if 50% or more of the respondents rate the recommended results in the eight testing conditions, ratings of 4-5 are "average" and usually signal that some improvement may be considered. Rating of 3 or low indicate serious problems. For example, a tester gives a rating: 5 on C1 (testing condition), he fills it in the first line, and then keeps testing and rating in C2~C8.

Table 7.1 Testing Form of Fashion Agent System

Testing condi-tion	Rank the recommendation results according to your preferences										Rating in seven-point scale
	1st Choice	2nd Choice	3rd Choice	4th Choice	5th Choice	6th Choice	7th Choice	8th Choice	9th Choice	10th Choice	
C1	2	1	9	3	7	4	5	6	10	8	
C2											
C3											
C4											
C5											
C6											
C7											
C8											

7.2 Hypothesis and precondition

When shopping, people usually have their personal preferences and constraints. It is realistic. For example, a lady wants to buy a dress, she knows that she never buys the clothes in "yellow" color, or/and she really hates the "Polo" brand. So we have this precondition:

- The user has his constraints in mind, such as brand = "Polo" or/and color ="Yellow". The Fashion Agent may not know these constraints.

In order to get more clear comparison results we make following assumptions:

- The user tests the system with the same query in the eight different testing conditions.
- The order of agents' recommended results is always from 1-10 according to the computed global similarity.

7.3 Evaluation

21 testers, who are the students and teachers in our department joined the experiment. We adopted two approaches to analyze and evaluate the 21 forms. One of the methods uses "Euclidean Distance" to measure the similarity between the recommended results provided by Fashion Agent and the evaluated orders by the testers. This will indicate the degree of satisfaction with the Agent's recommended results. Another method analyzes the rating results provided by users, so that we can get the performance of the agent in general users' view. Finally, we will compare these two evaluations and make some conclusions.

Method 1: The analysis of the agent's recommended results with Euclidean distance

Euclidean distance (as shown in Formula 7.1) is popular in distance measure, and it is used widely in similarity measures in Information Retrieval to measure two vectors' similarities [Schroeder & Noy, 2001]. In our case, the two vectors are the ranked sequences provided by Fashion Agent and users. One of vectors (B) is provided by the agent, which always has the format 1, 2, … , 10. Another vector (A) is provided by the users, which is the reorder of the agent's recommended results (e.g. 1, 2, 3, 4, 10, 6, 7, 8, 9, 5). Euclidean distance is used to measure these vectors' similarities, so that we can find the degree of users' satisfaction with the agent's recommended results.

$$ED_{AB} = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$$

Formula 7.1 Euclidean Distance

However, in our scenario, the elements in the two vectors not only show the numeric values, but also the ranked order, which have different importance for each element position. For example, there are two evaluation results: A = (1, 2, 3, 4, $\underline{10}$, 6, 7, 8, 9, $\underline{5}$) or A' = ($\underline{6,}$ 2, 3, 4, 5 $\underline{1,}$ 7, 8, 9, 10). The distances away from the Agent's sequence B = (1, 2, 3, … 10) is same (DE_{AB} = 7.07) if they are computed by Formula 7.1. However we find that the top four elements in vector A are matched perfectly with vector B, only the fifth and the last one have some distance compared to the Agent' results; on the other hand, the first and fifth element in the second vector A' are far to the agents' results. In human sense, the first result is closer to Agent's result than the second one. That means that previous elements are more important than the later ones. So different weights have to be added to each element in a vector to modify the formula. Also we know that the larger the distance, the smaller the similarity, and vice versa. As a result, a similarity formula (Formula 7.2) is defined through the normalized Euclidean distance as shown below. The similarity falls in $0 \le Sim_{AB} \le 1$.

$$Sim_{AB} = 1 - \frac{1}{N}\sqrt{\frac{\sum_{i=1}^{N} w_i (A_i - B_i)^2}{\sum_{i=1}^{N} w_i}}$$

Formula 7.2 Normalized Euclidean Distance

In Formula 7.2, Sim_{AB} is the similarity between two vectors; B_i is the order sequence provided by Fashion Agent, which is always (1, 2, … 10) in our scenario; A_i is the re-order sequence provided by the users; w_i represents the weight, which is given a value between 1 and 10 for each element. Weight 10 is given to the first order the user choose, 9 is given to the second one, until weight 1 is given the last one. N is the number of elements in a vector, which is the total number of recommended products. It equals to 10 in our case.

Also, through measuring and analyzing various cases, we define a *three-point scale* evaluated level depending on the normalized Euclidean distance (Sim_{AB}): Similarity ranges: 0~0.6 means the users are not satisfied by the recommended results, 0.6~0.8 means average satisfaction, and 0.8~1 means excellent match.

An example below shows how to compute the similarity between A (4, 1, 2, 3, 5, 6, 7, 8, 10, 9) and (1, 2, ... 10). The computed result is 0.85, which means the similarity between the recommended results and the users' requirement is 85%.

$$Sim_{AB} = 1 - \frac{1}{10}\sqrt{\frac{10*(4-1)^2 + 9*(1-2)^2 + 8*(2-3)^2 + 7*(3-4)\ \ + ... + 1*(9-1)^2}{10+9+...+1}}$$
$$= 0.85$$

In the example mentioned above, the similarities of vector A and A' to vector B are computed, as shown in Table 7.3. We find that the similarity of A to vector B, is higher than A', even through they have the same Euclidean distance. So this confirms the validity of the normalized similarity formula.

Table 7.2 An Example of Comparing Similarities With and Without Weights

Vector	Elements of the vector										Euclidean distance With Formula 1	Similarity With Formula 2
B	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*		
...												
A	1	2	3	4	10	6	7	8	9	5	7.07	0.826
A'	6	2	3	4	5	1	7	8	9	10	7.07	0.703
...												

All of the similarities between the agents' recommended results and the testers' re-ranked results in each testing conditions are computed. The analyzed results are shown in Figure 7.2, which presents the testers percent in a three-point similarity scale. The similarity of 37% testers is in the range 0.8~1, which is considered to be an excellent match, 53% is on average level with the similarity values from 0.6 to 0.8; only 10% of the recommended results fall in poor region with similarity values less than 0.6. This indicates that 90% of results is above average level, hence demonstrating the validity of our mechanism.

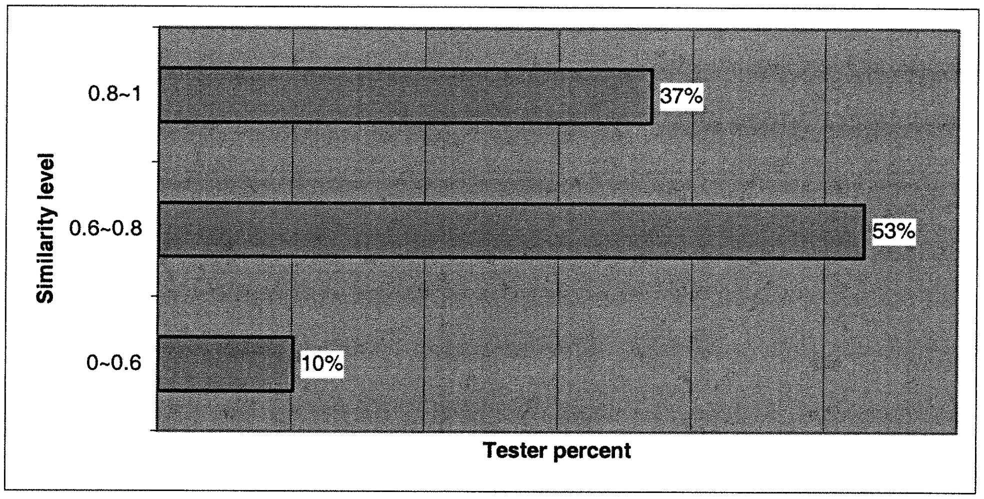


Figure 7.2 Tester Percent in Three Similarities Level

In order to analyze the recommend mechanism in detail, we especially list the mean similarities in each testing conditions as shown in Figure 7.3.

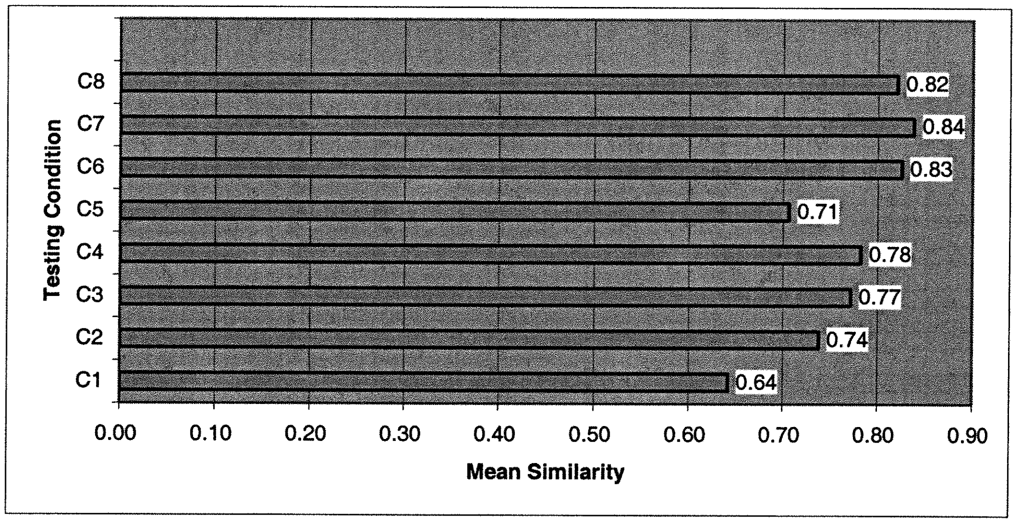


Figure 7.3 Mean Similarities in Each Testing Condition

The recommended results in C2~C5 have moderately higher similarities than in C1. This indicates that when the users provide their preferences or constraints to the agent, better matched results to users' requirement are achieved. This verifies the key role the short-term profiling or long-term profiling plays in the XMLFinder system. Moreover, we can see that there are strong higher similarities in C6 ~C8 than others. This means that when the users provide their short-term ***and*** long-term request to the agent, they can get the satisfied results. This also indicates that the combination of short-term and long-term profiling provides the more satisfying performance in our system.

Method 2: The analysis of rating of the recommended results

In Table 7.1, the testers give the ratings of the recommended results under eight testing conditions. We compute the average values. The mean value is shown in Figure 7.4, and the testers percent in three-point rating scale is shown in Figure 7.5 for each condition.

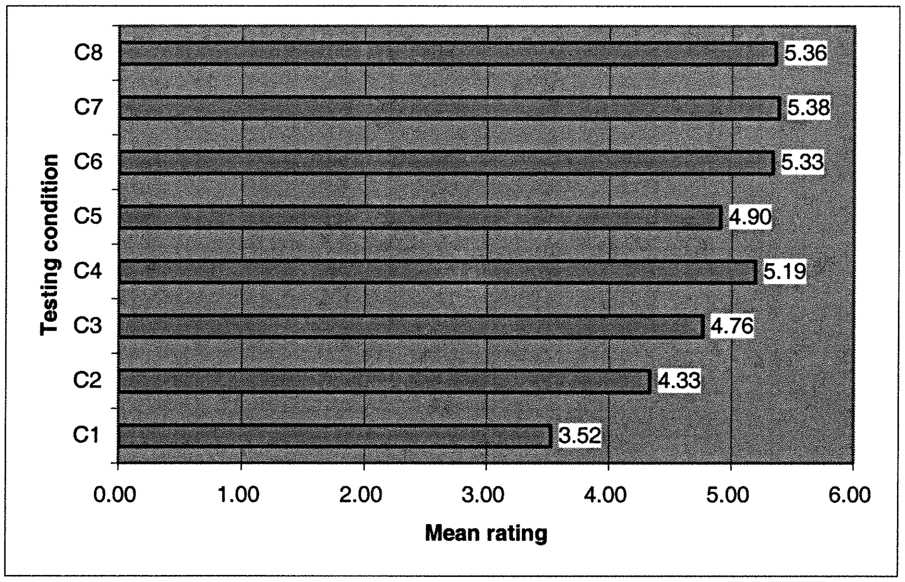


Figure 7.4 Mean Rating in each Testing Condition

As shown in Figure 7.4, we can see that the mean ratings in each testing condition, except in C1, are in average level (4~5), which reflects that the testers basically are satisfied with the recommended results. Also we find that the ratings in C2~C5 are higher than in C1, and lower than in C6~C8. It shows a similar conclusion to the one obtained in method 1. It further confirms that the short-term or long-term profiling is able to provide better performance than the simple searching (in C1) and the combination of short-term and long-term profiling could provide more satisfactory performance in this system. The evaluation results also show when the users interact with this system more than one time, their performance can be further improved or consistently satisfied.

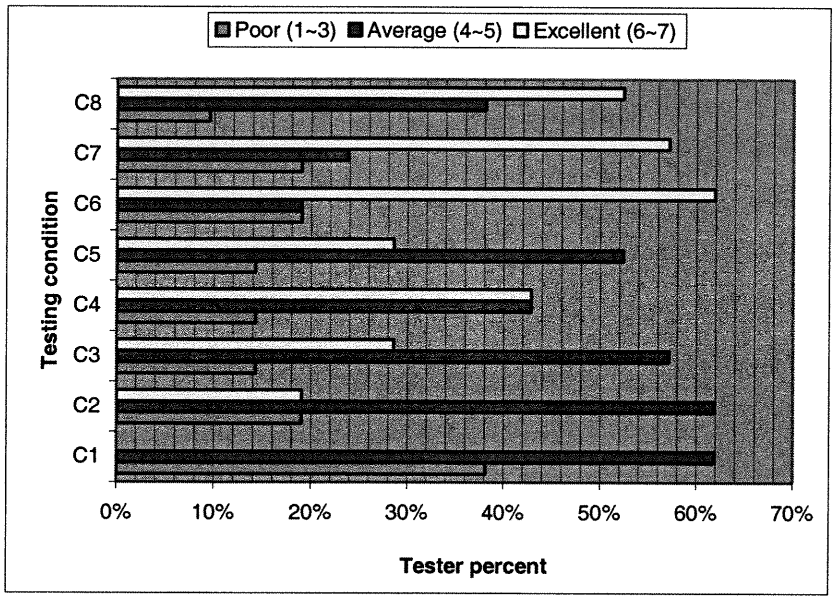


Figure 7.5 Tester Percent in Three-point Rating Scale in Each Testing Condition

However, as shown in Figure 7.5, we find that about 10% of the testers give poor rating in each testing condition, which indicates that testers are unsatisfied to some extent with presented the results. The main reason is that there are not enough products' cases in the database. This is due to the fact that one suitable query would get an excellent recommended result; on the other hand, another query, for which there is no similar product in the database may get a poor result. And then when we add the products' cases from 100 to 200, for example, by adding various colors and sizes to each brand clothes, a distinct improvement of the recommended results is obtained.

7.4 Summary of this evaluation

From method 1 and 2, the following conclusions can be made from this evaluation:

- The performance in C1 (simple searching) is inferior to other scenarios, but it is acceptable. This is due to the fact that this scenario implements the learning mechanism. The recommended results are ranked by the agent according to the user's basic requirement.
- Short-term profiling gives better performance than the simple searching.
- Long-term profiling also gives better performance than the simple searching. But its performance is not better than the short-term profiling.
- The combination of short-term and long-term profiling obtains the best performance.

In summary, we confirm that the approach -- the combination of short-term profiling and long-term profiling that we adopted in our system is efficient. Moreover, the feedback from the testers response shows that this system is easy to use. However, further improvement and testing are needed, such as adding more cases in database or finding more participants to evaluate the system.

Chapter 8

Conclusion and Future Work

XMLFinder is a shopping agent, which recommends products according to users' preferences and constraints to on-line shoppers, in order to reduce the "information overload" when shopping on Internet. It first uses SQL retrieval to obtain a set of broadly similar cases and then uses the nearest neighbor retrieval to rank those cases. If the recommended results do not exactly fit to the parameters entered, the customer increases the priorities of the most important parameters. Again, the system displays the ten best matches to the refined query, thus further improving the quality of the returned results. As an intelligent agent, XMLFinder, whose intelligent mechanism is based on CBR, can automate the profile-building process and finding a match to user's needs. The functions the agents could perform in this context are:

- It adjusts its own profiling mechanisms according to the user feedback
- It performs tasks on the user's behalf for searching and browsing various products, and comparing on products' attributes.
- It can train or teach the user to find products based on their preferences and constraints.

XMLFinder is constructed on a three-tier net architecture. Each tier has its own responsibilities; together they form a cohesive, flexible, and scalable shopping application, where XML takes an important role at each tier. XML is a useful communications language enabling large packets of formatted information to be exchanged thereby reducing network traffic. As a possible replacement to HTML it helps the web support intelligent applications. XSL is used to provide a clean separation between the content itself (XML) and the specific format required by a target application or output document.

CBR is a promising technology to develop intelligent agents. CBR systems are playing an increasingly important role in product selection and specification on-line. This work confirms that CBR is an effective methodology for e-commerce problem solving.

The main contributions of this work are:

- Development of an intelligent agent based on CBR technology, which uses an algorithm based on the combination of short-term and long-term profiling. It is verified that the approach can recommend products to on-line consumers efficiently.
- Discrete similarities' representation in XML document. According to the authors' knowledge, no other system finds discrete similarities to be stored and represented in XML. Our method provides a more effective way than storing data in a database or text file format since it is less costly than database storage and it allows the extraction of the attribute's meanings directly from XML tags.
- JavaServer Page (JSP) is used as a platform to develop an agent system. *A Fashion Agent,* as the protot7ype of XMLFinder is a web application built on Java Web server to give recommendations when consumers buy clothes. JSP scripts provide an alternative to creating CGI scripts that makes page development and deployment easier and faster. The programs include the programming logic knowledge sources implemented in an SQL and Java, and the data and knowledge sources implemented in XML document and an SQL database with the internal communications implemented using ODBC over TCP/IP.
- Providing performance comparison testing and evaluation to verify the validation of the approach we proposed. A normalized Euclidean distance is introduced to evaluate the users' degree of satisfaction by measuring similarities of the recommendation results.

Despite our effort to make XMLFinder a robust system, there are still areas for improvement. Some areas for future work are considered below:

- Since adaptation is the most important step in CBR, we need to add more solutions to make our system more intelligent. Adaptation is still a research issue in case-based reasoning that is also getting increasingly important in practice as the complexity of applications increases. However, all recent approaches lack a systematic approach to design considering the knowledge available in this domain.
- Collaboration with external agents, e.g. the use of existing shopping agents to gather more products information to provide for the users.
- Other more detailed studies are needed in different areas, such as adding more attributes, trying more complex query conditions, applying it in a larger context, maintaining similarity database automatically, etc.
- Making it to be a super assistant in a big shopping mal. It will combine several catalogs of different shops, so that the consumers can get the products' recommendations before they ask several different assistants in many shops. This will save a lot of consumers' time when they select their desired products.

Finally, there is an increased amount of information available on the Web. This information abundance increases the complexity of locating relevant information. Complexity drives the need for improved search and retrieval approaches. Intelligent agents are a way to improve search and retrieval as a personal assistant. The combination of the intelligent agent, CBR technology, and the universal capability of XML addresses the trust and competence issues of agents, enabling a further push towards the development of e-commerce systems.

References:

[Aamodt & Plaza, 1994] Aamodt, A. and Plaza, E. Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, Vol. 7(1). pp.39-59, *AI Communications*, 1994.

[Adam *et al*, 1999] Adam, N.R. Kogramaci, O. Gangopadhyay A. and Yesha, Y. Electronic Commerce --- Technical, Business, and Legal Issues. pp.5-8 *Prentice Hall*, 1999.

[Aha & Goldstone,1989] Aha, D.W. and Goldstone, R.L. Concept Learning and Flexible Weighting. *In Proceedings of a Workshop on Case-Based Reasoning, Pensacola Beach, FL,* pp.72-76, Morgan Kaufmann. 1989.

[Aïmeur, & Vezeau, 2000] Aïmeur, E. Vezeau M. Short-term Profiling for a Case-based Reasoning. *Machine Learning: ECML 2000, Lecture Notes in Artificial Intelligence 1810*, pp.23-30, 2000.

[Ajit, 2000] Ajit, S. XML RDBMS and OODBMS: Peaceful Coexistence? *XML Journal*, Vol. 1(2), pp.50-53, 2000.

[Bolin,1998] Bolin, S. E-commerce: A Market Analysis and Prognostication, *StandardView* Vol. 6(3): pp.97-105, 1998.

[Coppel, 2000] Coppel, J. E-commerce: Impacts and Policy Challenges, *Economics Department Working Paper*, No.252, 2000.

[Cunningham, 1998] Cunningham, P. CBR: Strengths and Weaknesses. *In Proceedings of 11th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems,* eds A. P. del Pobil, J. Mira & M. Ali, Lecture Notes in Artificial Intelligence 1416, Vol. 2: pp.517-523, Springer Verlag, 1998.

[Guttman & Maes, 1998] Guttman, R. and Maes, P. Agent-mediated Integrative Negotiation for Retail Electronic Commerce, *In proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98)*, Minneapolis, Minnesota, May 1998.

[Guttman *et al*, 1998] Guttman, R. Moukas, A. and Maes, P. Agent-mediated Electronic Commerce: A Survey, *Knowledge Engineering Review*, Vol. 13:3, June 1998.

[Hayes & Cunningham, 1999] Hayes, C. Cunningham, P. Shaping a CBR View with XML. *Third International Conference on Case-Based Reasoning, ICCBR-99* Seeon Monastery, Germany, pp.468-481, 1999.

[Hayzelden & Bigham, 1999] Hayzelden, A. & Bigham, J. Software Agents for Future Communication System. pp.5-6. Springer, 1999.

[Janet, 1993] Janet, K. Case-Based Reasoning. pp.150-152. Morgan Kaufmann Publishers, Inc. 1993.

[Kozierok & Maes,1993] Kozierok, R. and Maes, P. A Learning Interface Agent for Scheduling Meetings. *Proceedings of the ACM-SIGGHI International Workshop on Intelligent User Interfaces*, New York, ACM Press, pp. 81-88, January 1993.

[Ma & Aïmeur, 2001] Ma, Y. and Aïmeur, E. Intelligent Agent in Electronic Commerce XMLFinder, *10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Research*, Knowledge Media Networking Workshop, MIT, Cambridge, MA, June 2001.

[Maes, 1994a] Maes, P. Modeling Adaptive Autonomous Agents. Artificial Life Journal, edited by Langton, C. Vol. 1(1-2): pp.135-162, MIT Press, 1994.

[Maes, 1994b] Maes, P. Agents that Reduce Work and Information Overload. *Communications of the ACM,* Vol. 37(7):pp.31-40, 146, *ACM Press*, July 1994.

[Mitchell *et al*, 1994] T. Caruana, R., Freitag, D., McDermott, J. and Zabowski, D. 1994. Experience with a Learning Personal Assistant, *Communications of the ACM.* vol. 37(7) : pp. 80-91, 1994.

[North, 2000] North, K. XML and Database: There's Too Much Confusion. *XML Journal*, Vol. 1 (2): pp26-30, 2000.

[Nwana & Ndumu, 1998] Nwana, H.S. and Ndumu, D.T. BT Research Labs. A Brief Introduction to Software Agent Technology, Agent Technology --Foundation, Applications, and Markets, pp.31, Springer, 1998.

[Payne, Edwards & Green, 1997] Payne, T., Edwards, P. and Green, C.L. Experience with Rule Induction and K-nearest Neighbor Methods for Interface Agents that Learn. *IEEE Transactions on Knowledge and Data Engineering* 9, 2, pp.329-335, March 1997.

[Quin, 2000] Quin, L. Open Source XML Database Toolkit. Wiley, pp.166, 2000.

[Ricci & Avesani, 1995] Ricci, F. and Avesani, P. Learning a Local Similarity Metric for Case-Based Reasoning. *In Proceedings of Case-Based Reasoning Research and Development,* pp.301-312, Springer, 1995.

[Ricci & Avesani, 1999] Ricci, F. and Avesani, P. Data compression and local metrics for nearest neighbor classification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 21(4) : pp.380-384, 1999.

[Riesbeck & Schank, 1989] Riesbeck, C.K. and Schank, R.C., editors. Inside Case-Based Reasoning. pp.30-33 Lawrence Erlbaum Associates, Hillsdale, NJ. ISBN 0-89859-767-6, pp.30-33, 1989.

[Robert,1973] Robert A. Principles of Statistics and Probability. *Prindle, Weber & Schmidt, Incorporated*, pp.126-128, 1973.

[Schmitt *et al*, 2000] Schmitt, S. Maximini, R. Landeck, G. & Hohwiller, J] A Product Customization Module Based on Adaptation Operators for CBR System in E-Commerce Environments. *5th European Workshop, EWCBE 2000*, Springer, pp.504-516, 2000.

[Schroeder& Noy, 2001] Schroeder, M. & Noy, P. Multi-Agent Visualization Based on Multivariate Data, *In Proceeding of 5th International Conference on Autonomous Agents*, pp.85-91, 2001.

[Shardanand & Maes, 1995] Shardanand, U. & Maes, P. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. *Proceedings of the Computer-Human Interaction Conference (CHI'95),* Denver, CO, pp.210-217, May 1995.

[Vollrath *et al,* 1998] Vollrath, I., Wilke, W. and Bergmann, R. Case-based Reasoning Support for On-line Catalog Sales. *IEEE Internet Computing*. Vol. 2(4): pp.47-54, July-August, 1998.

[Watson & Gardingen, 1999] Watson, I. & Gardingen, D. A Distributed Case-based Reasoning Application for Engineering Sales Support. *In Proceeding of 16th International Joint Conference on Artificial Intelligence (IJCAI),* Vol. 1: pp.600-605, Morgan Kaufmann Publishers Inc. 1999.

[Wilke *et al*, 1998] Wilke, W. Bergmann, R. and Wess, S. Negotiation During Intelligent Sales Support with Case-Based Reasoning. *GWCBR'98, 6th German Workshop on Case-based Reasoning,* 1998.

URLs:

[AI_EC] AI for e-commerce, http://www.cs.umbc.edu/aiec/contents.shtml

[AI-cbr.org] Ai-cbr website, http://www.ai-cbr.org/theindex.html

[Andrew_Broad] Andrew Broad's Website – Case-Based Reasoning,http://www.cs.man.ac.uk/~broada/cs/cbr/cs3411.html

[Apache.org] Apache web site: http://www.apache.org

[Auctionbot.com] Auctionbot Web site: http://www.auctionbot.com/

[BargainFinder] BargainFinder Web site: http://bf.cstar.ac.com/bf

[BidFind] BidFind Web site: http://www.vsn.net/af/

[Bonzi] Bonzi.com Web site: http://www.bonzi.com/

[BottomDollar.com] BottomDollar Web site: http://www.bottomdollar.com

[XML-DB] XML : DB Web site: http://www.xmldb.org/index.html

[Clarke, 1998] Clarke, R. Electronic Data Interchange (EDI): An Introduction. 1998 http://www.anu.edu.au/people/Roger.Clarke/EC/EDIIntro.html

[CoCo, 1998] CoCo, J. Server-Side XML Taming the Tower of Babel, 1998. http://www.infoloom.com/gcaconfs/WEB/chicago98/jaenicke.htm

[CommerceNet] CommerceNet: http://eco.commerce.net/

[Cunningham *et al*, 2000] Cunningham, P. Bergmann, R. Schnitt, S. Traphoner, R. Breen, S. & Smyth, B. WEBSELL: Intelligent Sales Assistants for the World Wide Web, TCD-CS-2000-42. ftp://ftp.cs.tcd.ie/pub/tech-reports/reports.00/TCD-CS-2000-42.pdf

[Cykopahts.com] www.cykopaths.com

[DLJ.com] DLJ group: http://www.dlj.com/

[Firefly.com] Firefly Web site: http://www.firefly.com/

[Forrester.com] Forrester group: http://www.forrester.com/

[Gartner.com] Gartner group: http://www.gartner.com/

[IDC.com] IDC group: http://www.idc.com

[IE web site] Internet Explore Web site:
http://www.microsoft.com/windows/ie/default.htm

[JSP features] White Paper, JavaServer Pages™ Key Features
http://java.sun.com/products/jsp/keyfeatures.html

[JSP_XML] White Paper, Developing XML Solutions with JavaServer Pages Technology http://java.sun.com/products/jsp/html/JSPXML.html

[KeenanVision.com] Keenan Vision group: http://www.keenanvision.com/

[MSN–XML] MSN – XML Development Center page, http://msdn.microsoft.com/xml/

[MySimon.com] MySimon Web site: http://www.mysimon.com/

[OKIgroup] OKI group: http://www.okibusiness.com/

[Personalogic.com] Personalogic Web site: http://www.personalogic.com/

[RoboShopper.com] RoboShopper Website: http://www.roboshopper.com/

[Schussel, 2000] Schussel, G. Client/Server: Past, Present and Future.
http://news.dci.com/geos/dbsejava.html

[Shopfido.com] Fido Web site: http://www.shopfido.com/

[ShoppingExplorer.com] Shopping Explorer Website:
http://www.shoppingexplorer.com/

[SUN_JSP] Sun's JSP web site: http://java.sun.com/products/jsp/

[SUN_Servlet] Sun's Java Servlet web site: http://java.sun.com/products/servlet/

[SUN–XML] Sun – Java Technology and XML page, http://java.sun.com/xml/

[Tsang, 1993] Tsang, E. Foundations of Constraint Satisfaction Problems: A Survey. *Academic Press*, 1993. http://cswww.essex.ac.uk/CSP/edward/FCS.html

[Uversion.com] One-Click Computer ESP Bargain Agent Web site: http://oracle.uversion.com/shop/index.html

[VirtualLettingAgent] www.hookemacdonald.com

[VirtualOutlet.com] VirtualOutlet.com Website: http://www.virtualoutlet.com/

[W3.org] W3C – XML Web page, http://www.w3.org/XML/

[W3C_DOM] W3C – Document Object Model page, http://www.w3.org/DOM/

[XEDI.org] XEDI.ORG: http://www.xedi.org

[XMLS.com] XML Solution: http://www.xmls.com

[XML/XSL, 1999] White paper. XML/XSL: The Lifeblood of E-Commerce, Forrester Research, Inc. April 1999
http://www.itpapers.com/cgi/PSummaryIT.pl?paperid=994&scid=201

Appendix A

Investigation and Evaluation of XMLFinder system

In order to evaluate the validity of the approach we proposed in XMLFinder, we require your assistance to test this prototype system -- Fashion Agent. Try to understand the testing conditions and testing procedure before filling in the testing form. When you finish, please leave your evaluation form in Heron Lab (room 2374) or email to me. Thank you for your time and assistance.

1. Testing conditions

C1: Without constraints in the long-term profiling, and no preference in short-term profiling.

C2: Without constraints in the long-term profiling, selected preference ("better" in one attribute)

C3: Repeat C2 with same preference ("better" in this attribute again)

C4: Repeat C2, but with different preference ("better" in another attribute)

C5: With constraints in the long-term profiling, and no preference in short-term profiling (go to edit your constraints)

C6: With constraints in the long-term profiling, selected preference ("better" in one attribute)

C7: Repeat C6 with same preference ("better" in the same attribute again)

C8: Repeat C6, but with different preference ("better" in another attribute)

Note: *In this evaluation, the short-term profiling is the temporary preferences to a certain attribute; the long-term profiling here means the constraints of the values of one or more attributes provided by users.*

2. Operation and testing procedure

1. Open Fashion Agent system by going to my home page
2. Fill in a simple registration form to register to this system;
3. After giving your preference, you will get the recommended results. Then fill the following testing form;
4. Upon completion of the testing form, leave it in Heron lab, or email to me.

3. Testing form

It's better to download and print this form because you will fill it when interacting with this Fashion agent several times.

Testing condi-tion	Rank the recommendation results according to your preferences										Rating in seven-point scale
	1st Choice	2nd Choice	3rd Choice	4th Choice	5th Choice	6th Choice	7th Choice	8th Choice	9th Choice	10th Choice	
C1											
C2											
C3											
C4											
C5											
C6											
C7											
C8											

Your constraints: Brand: ____________ Color:____________
Size: ____________ Style: ____________

Your query: Brand:____________, Style:____________
Color:____________, Size:____________ Price:____________

Note: The seven-point scale ranges from 1 to 7, where 1= not satisfied, 4 = average, 7=excellent results.

Appendix A