

Université de Montréal

**PCFinder: An Intelligent Product Recommendation Agent
for Electronic Commerce**

Par:

Bin Xiao

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 ès sciences (M.Sc.)

en informatique

avril, 2003

© Bin Xiao, 2003



QA
76
U54
2003
v.040

Direction des bibliothèques

AVIS

L'auteur a autorisé l'Université de Montréal à reproduire et diffuser, en totalité ou en partie, par quelque moyen que ce soit et sur quelque support que ce soit, et exclusivement à des fins non lucratives d'enseignement et de recherche, des copies de ce mémoire ou de cette thèse.

L'auteur et les coauteurs le cas échéant conservent la propriété du droit d'auteur et des droits moraux qui protègent ce document. Ni la thèse ou le mémoire, ni des extraits substantiels de ce document, ne doivent être imprimés ou autrement reproduits sans l'autorisation de l'auteur.

Afin de se conformer à la Loi canadienne sur la protection des renseignements personnels, quelques formulaires secondaires, coordonnées ou signatures intégrées au texte ont pu être enlevés de ce document. Bien que cela ait pu affecter la pagination, il n'y a aucun contenu manquant.

NOTICE

The author of this thesis or dissertation has granted a nonexclusive license allowing Université de Montréal to reproduce and publish the document, in part or in whole, and in any format, solely for noncommercial educational and research purposes.

The author and co-authors if applicable retain copyright ownership and moral rights in this document. Neither the whole thesis or dissertation, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms, contact information or signatures may have been removed from the document. While this may affect the document page count, it does not represent any loss of content from the document.

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé

**PCFinder: An Intelligent Product Recommendation Agent
for Electronic Commerce**

Présenté par:

Bin Xiao

A été évalué par un jury composé des personnes suivantes:

Peter Kropf	Président-rapporteur
Esmâ Aïmeur	Directrice de recherche
Julie Vachon	Membre

Mémoire accepté le 11 août 2003

Résumé

Aujourd'hui, il y a beaucoup d'applications de commerce électronique (e-commerce) sur le web. Dans la plupart des sites web d'e-commerce, une lacune fréquemment rencontrée est le manque de service à la clientèle et d'outils d'analyse de marketing. Afin de surmonter ce problème, nous avons construit un agent intelligent basé sur le raisonnement à base de cas "Case-Based Reasoning" (CBR) et le filtrage collaboratif, que nous avons inclus dans notre système de recommandation de produits, appelé PCFinder.

Le système de recommandation de produits proposé a quatre caractéristiques principales. La première caractéristique concerne l'application de nouveaux algorithmes et méthodologies de CBR dans l'e-commerce. Nous proposons une heuristique pour représenter une mesure de similarité basée sur l'ordre ainsi qu'une méthode de modification de poids et adaptation. Nous supposons que les diverses valeurs de chaque attribut pour chaque cas puissent être mises dans un certain ordre et que les poids permettent d'indiquer l'importance relative des différents attributs si nous calculons la similarité globale entre deux cas. La mesure de similarité dynamique n'a pas besoin d'être stockée, réduisant ainsi les activités de maintenance et les frais exigés. La fonctionnalité de modification de poids permet à l'utilisateur de donner la priorité à des attributs selon leur importance. Le modèle d'adaptation divise les attributs de produit (ou interne) en attributs indépendants, attributs dépendants et attributs reliés. Dans ce modèle, les différents attributs s'adaptent de manières différentes.

La deuxième caractéristique concerne la combinaison de technologies multiples. Le noyau du système est un agent intelligent. En outre, le CBR et les techniques de filtrage de collaboration ont été utilisés pour rendre cet agent plus efficace et plus

performant. Finalement, des techniques d'analyse de "clusters" ont été employées pour aider l'agent intelligent à grouper les clients selon leurs profils à long terme; ceci permet à l'agent d'analyser plus tard les profils d'utilisateurs (attributs externes) et de fournir des suggestions sur les caractéristiques du produit.

La troisième caractéristique est basée sur une méthode pour construire des systèmes de recommandation de produit (PRS). Nous décrivons tous les aspects des PRS: de l'architecture aux algorithmes et des technologies appliquées aux implantations.

La dernière caractéristique est la disponibilité d'un agent qui construit des graphiques basés sur l'analyse des "clusters" pour permettre aux personnels de gestion de détecter les tendances du marché. Cet agent analyse les enregistrements des ventes et des commandes précédentes, combinant les données des profils de l'utilisateur (attributs externes) avec des descriptions de produits (attributs internes).

Abstract

There are many electronic commerce (e-commerce) applications on the web today. A common shortcoming is the lack of customer service and marketing analysis tools in most e-commerce web sites. In order to overcome this problem, we have constructed an intelligent agent based on Case-Based Reasoning (CBR) and collaborative filtering, which we have included in our product recommendation system, called *PCFinder*.

The proposed product recommendation system has four main characteristics. The first one is applying the novel algorithms and methodologies of CBR for an e-commerce application. We propose a heuristic to represent an Order-Based Similarity Measure, together with the methods of weight modification and adaptation. We assume that the various values of each specific attribute in each case could be put in a certain order and the weights provide some indications of the relative importance of the different attributes when we calculate the global similarity between two cases. The dynamic similarity measure does not need to be stored, reducing the required maintenance activities and expenses. The weight modification feature allows the user to prioritize attributes according to their relevant importance. The adaptation module divides the product (or internal) attributes into independent attributes, dependent attributes and related attributes. In this module, different attributes adapt in different ways.

The second feature of the proposed recommendation system is that it combines multiple technologies. The core of the system is an intelligent agent. In addition, CBR and collaborative filtering techniques were employed to make this agent more efficient and effective. Finally, clustering analysis techniques were used to help the intelligent agent group customers according to their long-term profile; this allows the agent to later analyze the user profiles (external attributes) and provide some suggestions of items (internal attributes) of the product.

The third feature is the introduction of a method for constructing product recommendation systems (PRS). We describe all aspects of the PRS: from architecture to algorithms and from applied technologies to implementations.

The last feature is the provision of a graphic-building wizard based on clustering analysis to allow the management staffs to detect market tendencies. This wizard analyzes records of previous sales and orders, combining users' profiling data (external attributes) with product descriptions (internal attributes).

Table of Contents

Résumé	I
Abstract	III
Table of Contents	V
List of Tables	VIII
List of Figures	IX
Préface	XII
Preface	XIII
Acknowledgement	XIV
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Organization	2
Chapter 2 State of the Art – Agent-based Systems in E-Commerce	4
2.1 Consumer Buying Behavior Model	4
2.2 The Role of Agents in E-Commerce Systems	6
2.3 Technologies Used in Product Recommendation Systems	12
Chapter 3 Technologies Used in PCFinder	18
3.1 Agent Technology	18
3.1.1 Introduction to Agents	18
3.1.2 Classification of Agents	19
3.2 Case-Based Reasoning	23
3.2.1 The CBR Concept: Using Old Solutions to Solve New Problems	24
3.2.2 Similarity Measures	25
3.3 Collaborative Filtering	28
3.4 Clustering Techniques	30
3.4.1 Clustering	30

3.4.2 A Categorization of Major Clustering Methods	31
3.5 XML Technology	32
Chapter 4 Architecture of PCFinder	35
4.1 Two-tier versus Three-tier Architectures	35
4.2 Architecture of PCFinder	37
Chapter 5 Methodologies and Algorithms	41
5.1 Dynamic Order-Based Similarity Measure	41
5.2 Weight Modification	44
5.3 Adaptation	46
5.4 Profile and Collaborative Filtering	52
5.4.1 Long-term and Short-term Profiles	52
5.4.2 Profiling and Collaborative Recommendation	53
5.5 Clustering Analysis	54
Chapter 6 Implementation of PCFinder	59
6.1 Run-time Environment and Developing Tools	59
6.2 Main Class Diagrams of PCFinder	62
6.2.1 Class Diagram of Management Module	62
6.2.2 Class Diagram of Search Module	65
6.2.3 Class Diagram of Adaptation Module	66
6.2.4 Class Diagram of Clustering Analysis	67
6.3 Main Scenario Diagrams of PCFinder	69
6.3.1 Unregistered Users' Shopping Behavior Scenario	69
6.3.2 Registered Users' Shopping Behavior Scenario	69
6.3.3 The Administrator's Management Behavior Scenario	70
6.4 User Interface of PCFinder	71
6.4.1 Starting up PCFinder	71
6.4.2 Customer Entry	72

6.4.2.1 Registration and Constraints Suggestion	72
6.4.2.2 Product Recommendation	75
6.4.2.3 Browsing Function and Password Reminder	82
6.4.3 Administrator Entry	83
6.4.3.1 Cluster Analysis	84
6.4.3.2 Graphical Analysis	85
6.4.3.3 Maintenance	87
Chapter 7 Experiment and Evaluation	90
7.1 Experimental Process and Methods	90
7.2 Evaluation Methods	92
7.3 Evaluation Results	96
Chapter 8 Conclusions and Future Work	102
References	105

List of Tables

Table 2.1 Online shopping systems vs. the CBB model stages	7
Table 2.2 Technologies used in various agent-based systems	13
Table 5.1 Local similarity measures for the “processor speed” attribute	43
Table 5.2 Local similarity measures for the “brand” attribute	44
Table 5.3 The system state before the “brand” critique	45
Table 5.4 The system state after the “brand” critique	45
Table 6.1 Attributes and values of the notebook computers	60
Table 7.1 Test form for Case 1	91
Table 7.2 Test form for Case 2 and Case 3	91
Table 7.3 Result of Step 1	97
Table 7.4 Result of Step 1 vs Step 2	98
Table 7.5 Result of Step 2 vs Step 3	98
Table 7.6 Result of Case 1 vs Case 2	99
Table 7.7 Result of Case 1 vs Case 3	99

List of Figures

Figure 2.1 Consumer Buying Behavior model	5
Figure 3.1 Typology based on Nwana's primary attribute dimension	20
Figure 3.2 Steps of the CBR process	25
Figure 3.3 Representation of cases with n attributes	26
Figure 4.1 Three-tier architecture of PCFinder	37
Figure 4.2 Management Module	38
Figure 4.3 Search Module	38
Figure 4.4 Adaptation Module	39
Figure 4.5 Clustering Analysis Module	39
Figure 4.2 The transformation process from XML to HTML	40
Figure 5.1 Relationship between independent, dependent and related attributes ...	49
Figure 5.2 The framework of the adaptation module	51
Figure 5.3 Quantity of sales based on purchase records versus processor speed ...	55
Figure 5.4 Quantity of sales based on purchase records versus brand	56
Figure 5.5 The representation of three attributes in a case	57
Figure 5.6 The correlation between external and internal attributes	58
Figure 6.1 Class diagram of management module	64
Figure 6.2 Class diagram of search module	66
Figure 6.3 Class diagram of adaptation module	67
Figure 6.4 Class diagram of clustering analysis module	68

Figure 6.5 Scenario diagram of unregistered users' shopping behavior	69
Figure 6.7 Scenario diagram of registered users' shopping behavior	70
Figure 6.8 Scenario diagram of the administrator's management behavior	70
Figure 6.8 Main page of the online notebook computer store	71
Figure 6.9 Login page of the online notebook computer store	72
Figure 6.10 Contact information form	73
Figure 6.11 Profile information form	74
Figure 6.12 Long-term constraint form	74
Figure 6.13 Basic configuration form	75
Figure 6.14 Advanced configuration form	76
Figure 6.15 The five most similar computers	77
Figure 6.16 Interaction with the customer: add to cart or not?	77
Figure 6.17 Shopping cart	78
Figure 6.18 Before weight modification	80
Figure 6.19 After weight modification	80
Figure 6.20 After adaptation of the result	81
Figure 6.21 Buying session finished	81
Figure 6.22 Browsing by processor speed	82
Figure 6.23 Password retrieval – step one	83
Figure 6.24 Maintenance main page	84
Figure 6.25 User profile form	85
Figure 6.26 Configuration after clustering analysis	85
Figure 6.27 User interface of graphic-building wizard	86
Figure 6.28 Statistical diagram generated by the graphic-building wizard	87
Figure 6.29 Browsing selection of the purchase records	88
Figure 6.30 Browsing result of the purchase records	88
Figure 6.31 User interface of significant case maintenance tool	89

Figure 7.1 Tester percentage on three-point scale	97
Figure 7.2 Comparison of the satisfaction rates for the three steps	100
Figure 7.3 Comparison of the satisfaction rates for the three test cases	101

Préface

Le travail de cette thèse est basé sur, mais pas limité à :

PCFinder : An intelligent Product Recommendation Agent for e-Commerce [Xiao et al., 2003], une étude qui a été acceptée comme un exposé régulier à présenter à la *2003 IEEE Conference on E-Commerce (CEC'03)* qui s'est tenue du 24 au 27 juin 2003, à Newport Beach en Californie, et

Personalizing Product Recommendation in E-Commerce [Aïmeur et Xiao, sous presse], une deuxième étude qui sera présentée à la *IFIP (International Federation for Information Processing) Conference on E-Commerce, E-Business & E-Government - I3E2003*, à São Paulo en Guarujá (Brésil), du 21 au 24 septembre 2003.



Preface

The work on this thesis was based on, and not limited to, those of:

PCFinder: An intelligent Product Recommendation Agent for e-Commerce [Xiao et al., 2003], a study that has been accepted as a regular paper for presentation at the 2003 IEEE Conference on E-Commerce (CEC'03) which held in Newport Beach, California, during June 24-27, 2003; and

Personalizing Product Recommendation in E-Commerce [Aïmeur & Xiao, in press], a second study that is to be presented at The IFIP (International Federation for Information Processing) Conference on E-Commerce, E-Business & E-Government - I3E2003, São Paulo, Guarujá, Brazil, September 21-24, 2003.



Acknowledgement

I wish to sincerely thank my research advisor, Dr. Esma Aïmeur, for her kind guidance and support along the development of this thesis. I thank her for providing me with this opportunity to benefit from her knowledge.

I also take this opportunity to express my sincere thanks to Dr. Gilles Brassard for his valuable suggestions and comments.

I would like to thank the members of the HERON and GRITI who have helped and encouraged me in my work. I would like to thank José Manuel Fernandez for giving me a hand whenever I met difficulties. Thanks are also due to many volunteers for testing the recommendation system and providing helpful comments. I am also indebted to Jennifer Thibault for her valuable proofreading.

At last, I dedicate this thesis to my dear wife Yu Gu for her encouragement and patience throughout my thesis writing and to my son Ziyu Xiao who provided an additional and joyful dimension to my life mission. Without them, it would have been much more difficult for me to reach this point.

Introduction

1.1 Motivation

Electronic commerce is steadily becoming more important in changing the way people exchange products and services. It provides a convenient and easy way for both the consumer to buy things and the merchant to sell things. However, some problems remain to be solved if people wish to take complete advantage of this new paradigm. One of these problems is the lack of customer service and marketing analysis featured in e-commerce applications. Currently, product support offered by most organizations to their Internet customers is of comparatively poor quality, if it exists at all. Certainly, most organizations' web sites offer their customers the possibility to query (as opposed to physically examine) the available products using online catalogues, textual search engines or database interfaces. These tools, however, require the user to make an effort during the interaction phase. Often, these tools are not enough for the consumer (i.e. the Internet user) if the number of products is substantial, if the products are similar to each other, or if the consumer does not know the domain very well. Customers who feel completely overwhelmed by the number of choices may simply leave the site and never return [Aïmeur & Vézeau, 2000].

One of the solutions to this problem is to use *Product Recommendation Systems (PRS)* that proactively suggest products to the users according to their specified preferences and requirements. *PRS* contribute to increase customer satisfaction, therefore to enhance brand recognition and improve market performance for the organization. One

of the possible technologies for the conception of recommendation systems is *Case-Based Reasoning (CBR)* [Aamodt & Plaza, 1994]. The purpose of this sub-domain of artificial intelligence is to conceive knowledge-based systems that when faced with a new problem, reuse and adapt the solutions to similar problems in the past [Aïmeur & Vézeau, 2000].

1.2 Organization

This thesis introduces a method based on CBR and collaborative filtering for creating PRS and is organized into the following chapters.

Chapter 2 provides a brief overview of the *Customer Buying Behavior (CBB)* model [Guttman *et al.*, 1998; Guttman & Maes, 1998; and Moukas *et al.*, 1998], which is used to categorize various agent-based systems in e-commerce.

Chapter 3 reveals the different technologies used in *PCFinder*, whose purpose is to help an online computer shop suggesting the most appropriate products to its clients. It applies agent technology, case-based reasoning, collaborative filtering, clustering techniques and XML technology.

Chapter 4 depicts the advantage of three-tier architecture and the architecture of our prototype, *PCFinder*. The first tier provides an interface for the user to access data. The third tier contains the data sources (i.e. the company database). The middle tier is in charge of retrieving the request from the first tier and the data from the data source. After processing this data, it sends feedback to the first tier or the third tier.

Chapter 5 presents the methodologies and algorithms used in *PCFinder*. It describes

in turn dynamic Order-Based Similarity Measure, weight modification, adaptation, profile and collaborative filtering, and finally clustering analysis.

Chapter 6 describes the implementation of *PCFinder*. After introducing the run-time environment and the development tools, it describes main class diagrams, scenario diagrams and the user interface in detail.

Chapter 7 delivers an experiment and evaluation, a process that is based on the collection of user feedbacks. Two methods are applied in this evaluation. The first method applies modified “*Euclidean Distance*” to evaluate the similarities between the recommended results provided by *PCFinder* and the evaluated results provided by the users. A second method uses inference from large samples to measure the degree of user satisfaction.

Finally, Chapter 8 draws conclusions about our methods and proposes some future directions.

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

Electronic commerce is booming with increasing accessibility to the Internet in virtually every corner of the world. In the new generation of e-commerce, agent-based systems are becoming an attractive paradigm. Agents have demonstrated their tremendous potential in conducting various tasks in e-commerce, such as searching, buying and selling products, etc. Most of the tasks in the *Consumer Buying Behavior (CBB)* model can now be facilitated or automated by the use of agent technology.

2.1 Consumer Buying Behavior Model

The *CBB* model [Guttman *et al.*, 1998; Guttman & Maes, 1998; and Moukas *et al.*, 1998] is a simplified model of the decision process involved in a consumer making a purchase. There are six different stages: *Need Identification*, *Product Brokering*, *Merchant Brokering*, *Negotiation*, *Payment and Delivery*, and *Product Services and Evaluation*. This model simplifies a more complex behavior in which the stages are not discrete entities; in reality, these stages can overlap or even be concurrent, and the decision process can be iterative. This model can be illustrated as shown in Figure 2.1 [Turban *et al.*, 2002]. Although it is a simplified model, it provides an important tool to determine in what parts of the customer shopping process agent-based e-commerce

systems can be applied [Guttman *et al.*, 1997; Ripper *et al.*, 2000].

Each of these stages has its own specificities, but they agree on a set of fundamental stages to represent the consumer buying process. The stages defined by the *CBB* model are:

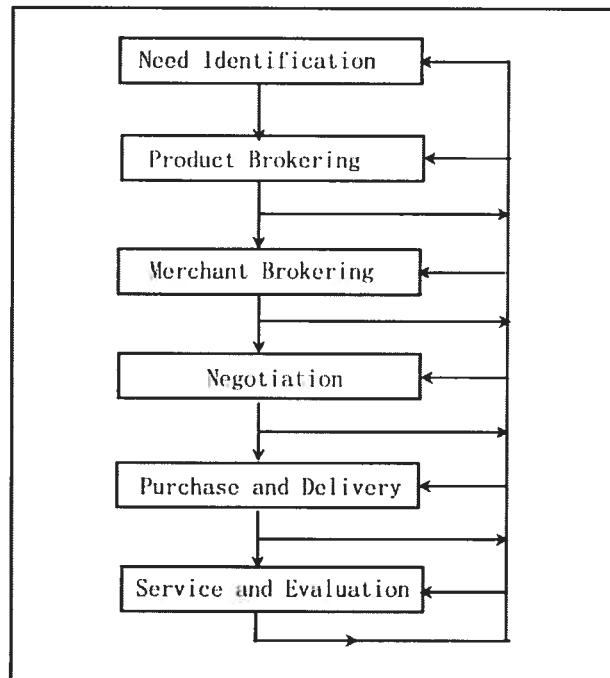


Figure 2.1 Consumer Buying Behavior model [Turban et al., 2002]

- *Need Identification*: In this stage, the customer recognizes the need to make a purchase. It is sometimes also called *Problem Recognition*.
- *Product Brokering*: In this stage, the consumer decides what to buy by retrieving information about the products. After evaluating a set of different products, he tries to identify which one would satisfy his needs.
- *Merchant Brokering*: In this stage, the customer determines from whom to buy the product. As a result of the previous stage, the customer already knows what he wants. The decision process includes the evaluation of merchant alternatives based on a set of criteria, such as price warranty, availability, reputation, etc.

-
- *Negotiation*: In this stage, the customer determines the terms of the transaction. Many of the traditional models do not identify this stage explicitly, but isolating this process is very useful for categorizing agents according to their roles.
 - *Payment and Delivery*: This stage sometimes signals the termination of the negotiation stage. The customer arranges the payment and delivery, buys warranties, etc.
 - *Product Services and Evaluation*: This stage includes any post-purchase product services or customer services. Also, the customer evaluates his satisfaction with the overall buying experience and his purchase decision.

2.2 The Role of Agents in E-Commerce Systems

With the increase in the number of customers, products, vendors and in the amount of information in general, it becomes more difficult if not impossible to match each individual customer with the best product available. Intelligent software agents offer a practical solution to handle this information overload. They help customers conduct routine tasks, search and retrieve information, support decision-making, and act as domain experts [Turban *et al.*, 2002].

Intelligent agents employed in e-commerce product recommendation systems suggest products to their customers and provide related information to help them decide which product to buy, from whom to buy.

Agents in e-commerce systems can be classified according to what stage in the *CBB* model they assist. Although these systems generally concentrate on one particular stage, it becomes more and more common for systems to work on two or more stages, such as *COGITO* [COGITO, 2001], which assists the stages of *Product Brokering*,

Merchant Brokering and *Negotiation*. These new agent systems try to get closer to the real buying experience, in which the stage transitions are not completely discrete [Ripper *et al.*, 2000].

Table 2.1 Online shopping systems vs. the CBB model stages

	Need Identification	Product Brokering	Merchant Brokering	Negotiation	Payment & Delivery	Service & Evaluation
Amazon	✓				✓	
Expedia	✓					
Firefly		✓				
Personologic		✓				
Webdoggie		✓				
NewsWeeder		✓				
Jola		✓				
CoCoA		✓				
Quickstep		✓				
BargainFinder			✓			
Jango			✓			
Kasbah				✓		
Tetc-@-tetc	✓	✓	✓	✓	✓	✓
COGITO		✓	✓	✓		
Firepond						✓
ITR		✓				

Table 2.1 introduces some of the agent systems available so far that focus on automating or assisting the *CBB* stages. A logical way to classify the agents in e-commerce systems is by relating them to the *CBB* stages. The rest of this section expounds the agent-centric stages of the *CBB* model and related Web applications.

Need Identification

Agents can assist the customers by supplying product information and stimuli during this stage.

For example, *Amazon* [Amazon] provides an agent to notify the customers when a new book in their area of interest arrives [Turban *et al.*, 2002]. *Expedia* [Expedia] notifies customers about low airfares to their desired destination whenever they become available [Turban *et al.*, 2002]. This stage is often combined with product brokering, since some agents that find products also compare prices based on the customer's criteria.

Product Brokering

After identifying the need for a product, the customer decides what product to buy. A variety of agent systems exist to help customers find the right product.

Firefly (now a part of Microsoft .Net) [Turban *et al.*, 2002] is the pioneering agent in this category. Customers participating in the Firefly program identify themselves using a “passport” when they visit participating sites, and the Firefly system recommends products/services to them. AOL-owned *Personalogic* [Personalogic] makes product recommendations based on the prioritization of attributes specified by the customer, such as price and delivery time. *Webdoggie* [MIT Media Lab] is a system that uses collaborative filtering to help people find web pages that are likely to interest them. *NewsWeeder* [NewsWeeder; Lang, 1995] recommends news articles to users, based on monitored reading interests and/or their rating of past articles on a scale of 1 to 5.

Jola [Jola; Bergmann *et al.*, 2002] assists the product brokering stage. *Jola* helps the customers to identify their target products with the help of a customization service although they are not familiar with all the modification possibilities and constraints of some products.

CoCoA [CoCoA; Aguzzoli *et al.*, 2002], which stands for Compilation Compiler

Advisor, also works on the product brokering stage. It allows the customer to create a music compilation using a repository of sound tracks and a case base of compilations. By selecting the recommendation option, the customer can access a list of tracks that the system considers appropriate for the current partial compilation. The customer can also reuse or modify compilations that previous customers have created in the past by consulting the database of compilations.

Quickstep [Middleton *et al.*, 2001] assists the user to find new published research papers in their general field of interest and old relevant papers according to the user's feedbacks. Any feedback of these recommendations is recorded when the user looks at them. A set of papers is recommended based on correlations between user interest profiles and classified paper topics.

ITR (Intelligent Travel Recommender) [Ricci *et al.*, 2002] is a web-based recommender system designed to help a user to select travel products (e.g. a hotel, a museum, a climbing school etc.) and build a travel package that includes location, accommodation, and organized activities.

Merchant Brokering

Once the customer has decided what product to buy, he compares different merchant alternatives to determine where to buy this product.

BargainFinder [Krulwich, 1996] was the pioneering agent in this category. Given the title of a music CD, it would return the price of the CD at various online music retailers. *Jango* [Jango] is similar to *BargainFinder*, but it includes more product categories and also provides product reviews. With *Kasbah* [MIT Media Lab], users who want to buy or sell a product assign the task to an agent that proactively seeks buyers or sellers on their behalf.

Negotiation

During this stage, the customer determines the terms of a transaction (e.g. price). The advantage of dynamic pricing is that the price is set by the marketplace, rather than by the seller. In a fixed price situation, if the seller demands a price too high, sales will suffer; if the price is set too low, profits will be affected. With dynamic pricing, limited resources tend to be distributed more fairly [Maes *et al.*, 1999].

Kasbah agents can negotiate with each other according to specific strategies assigned by their creators. It provides buyers with one of three negotiation “strategies” - anxious, cool-headed, and frugal - corresponding to a linear, quadratic, or exponential function respectively for increasing its bid for a product over time (similar functions exist for selling agents) [Maes *et al.*, 1999].

Tete-@-tete [MIT Media Lab] agents negotiate a number of different parameters: price, warranty, delivery time, service contracts, return policy, loan options, and other value-added services. This system integrates the six stages of the *CBB* model.

COGITO [COGITO, 2001] is a system that incorporates three stages: product brokering, merchant brokering and negotiation. It aims at a system that is not only reactive to some customer requests, but also proactive and capable of engaging in a goal-directed conversation with the customer. Hence, it can specify the overall goal of a complex task the customer intends to fulfill, clarify uncertainties in the understanding of the goal/task and negotiate the best strategy, and monitor and interpret the customer’s behavior (actions and information conveyed during interaction).

Payment and Delivery

Agents are used extensively during the actual purchase process, including arranging payment and delivery with the customer. The payment and delivery of a product can either signal the termination of the negotiation stage or occur sometime afterwards (in either order). In some cases, the available payment options (e.g., cash only) or delivery options may influence product and merchant brokering [Guttman *et al.*, 1998].

The payment and delivery stage is supported by using online payment systems. Actual online delivery of the good is possible if the product is in electronic form. Tangible goods are shipped the traditional way by the sellers themselves [Vetter & Pitsch, 1999].

For example, if a customer makes a mistake while completing an electronic payment form, the agent will point it out immediately. When a customer buys stocks, for example, the agent will tell the customer when a stock he wants to buy is not marginable, or when the customer does not have sufficient funds. At *Amazon.com*, delivery options are organized by agents, and the total cost is calculated in real-time [Turban *et al.*, 2002].

Service and Evaluation

In this stage, customers can receive post-purchase service and send feedback to agents. This post-purchase stage involves product service, customer service, and an evaluation of the satisfaction of the overall buying experience and decision. The nature of this stage (and others) depends upon for whom the product was purchased [Guttman *et al.*, 1998].

Agents can be used to facilitate after-sale service. For example, automatic answering agents that send e-mail responses are usually productive in answering customer queries. Agents can also monitor automobile usage and notify customers when it is time to take a car in for periodic maintenance. Agents that facilitate feedback from customers are also useful [Turban *et al.*, 2002].

The service and evaluation stage is addressed by adding a feedback functionality through which the users can give feedback to the market (evaluating the quality of a deal, including partner reliability and related matters, such as seller's reputation) and the seller directly (providing feedback about the quality of the good and the service) [Vetter & Pitsch, 1999].

Firepond [Firepond] is one of examples that provide online customer assistance solutions to help companies more profitably acquire and retain customers [Turban *et al.*, 2002].

2.3 Technologies Used in Product Recommendation Systems

There exist several ways to classify recommendation techniques [Resnick & Varian, 1997; Schafer *et al.*, 1999; and Terveen & Hill, 2001]. Authors involved in this discussion are not focused on the type of interface or on how the users interact with the recommendation agent; rather, they are focused on the sources of the data on which the recommendation is based and how this data is put into use [Burke, 2002].

According to the sources of data on which recommendation is based and the way in which that data is put into use, the majority of product recommendation systems is developed using *content-based* [Moukas, 1997], *collaborative-based* [Shardanand & Maes, 1995], *constraint-based* [Kumar, 1992] filtering or *knowledge-based* [Burke,

2000] methods as underlying technologies. Recently, *Case-Based Reasoning* (CBR) [Bergmann *et al.*, 2002], as a category of knowledge-based methods, has become a promising technology in agent-based e-commerce systems [Guttman *et al.*, 1998; Ma & Aïmeur, 2001]. We will briefly introduce these technologies (see Table 2.2).

Table 2.2 Technologies used in various agent-based systems

	Content-Based	Collaborative-Based	Constraint-Based	Case-Based
Jola			✓	✓
CoCoA		✓		✓
COGITO	✓	✓		
Quickstep	✓	✓		
NewsWeeder	✓			
Personalogic			✓	
ITR		✓		✓

A *content-based filtering* [Moukas, 1997] system selects items by matching the content of the item descriptions in the catalogue with the content of the customer preferences and requirements. The techniques applied in content-based filtering vary in complexity. One of the simplest techniques is the keyword-based search that consists of comparing different combinations of keywords and finding the match. For example, document recommendation systems, such as the newsgroup filtering system *NewsWeeder*, use the words in the text as features to recommend documents. A more advanced form of content-based filtering is that based on extracting semantic information from the content of the document.

One of the most familiar, widely implemented and mature techniques used in product recommendation systems is *collaborative-based filtering* [Shardanand & Maes, 1995]. Using this technique, people are statistically grouped according to common interests or according to their profiles. Collaborative-based filtering systems create profiles for users by collecting information on items they liked or disliked. Each profile represents the long-term interests of an individual or a group. To recommend an item to a user,

collaborative-based filtering systems first compare the user's profile with the profile of other users using statistical methods and then recommend items that people with similar profiles have appreciated. (Note the profiles are defined as *external attributes*, while the items are defined as *internal attributes*.)

Constraint-based [Kumar, 1992] techniques, like content-based approaches, use the features of an item to determine its relevance. These techniques require that the problem and solution space be formulated in terms of variables, domains, and constraints. Once the problem is formulated in this way, a number of general purpose and powerful *constraint satisfaction problem (CSP)* techniques can be applied to find a solution. *Personallogic* is an example of a constraint-based filtering system. It allows customers to narrow down on the number of products by guiding them through a large product feature space in order to find the products that best suit their needs. After the customer specifies constraints on product features, the system filters out unwanted products within a given domain.

Another popular approach is that used by *knowledge-based systems* [Burke, 2000]. These systems apply knowledge about users and products to generating a recommendation, reasoning about what products meet the user's requirements. This approach can be seen as an interactive way to guide the client through the product classification tree. A particular category of knowledge-based systems employs *Case-Based Reasoning*. To solve a new problem, these systems use the solutions to similar problems in the past. The *Case-Based Reasoning* process typically consists of four steps [Aamodt & Plaza, 1994]: the search for a similar case in the case base, the adaptation of the old solution to the new problem, the evaluation of the proposed solution, and finally the addition of the new case to the case base. For the last few years, *Case-Based Reasoning* has been used in e-commerce applications to help the client find products and services.

Hybrid Recommendation Systems combine two or more recommendation techniques to obtain better performance with fewer drawbacks. Most commonly, collaborative filtering is combined with some other techniques, such as content-based filtering or case-based reasoning, in order to avoid the problem of collaborative recommendation, which casual users cannot receive the full benefits in collaborative filtering system [Burke, 2002].

Jola is an intelligent virtual sales agent. The case-based retrieval component uses a product model based on attributes. *Jola* also uses constraint-based techniques. For example, even once the right product has been found, the customer may have further specifications to make, such as accessories. For some components, the accessories are even indispensable [Bergmann *et al.*, 2002].

CoCoA aims to extend the standard collaborative filtering approach to enable its use in a CBR loop in the context of compositional recommendation systems. [Aguzzoli *et al.*, 2002].

COGITO combines both content-based and collaborative filtering. In this system, user profiles are generated based on content features extracted from documents that users find relevant. The system also clusters users into virtual communities according to their expressed taste to then provide recommendations to each group. Generally collaborative filtering is based purely on users' opinions, so it is especially useful in taste-based domains such as books, music, movies, or TV. Machine learning techniques are used to associate user features with tastes and purchases.

Quickstep is also a hybrid recommendation system combining content-based and collaborative filtering techniques. As both web pages and user interests are dynamic in nature, catalogues, rule bases and static user profiles would quickly become outdated. Therefore *Quickstep* is well suited for these problems.

ITR integrates Case-Based Reasoning with interactive query management to create a system that understands a user query, suggests or answers related questions, and gives an approximate response. A new collaborative approach is used to analyze the traveler's behavior and extend the recommender systems [Ricci *et al.*, 2002].

Furthermore, some manufacturers work for the configuration of personal computers (which assist the product brokering stage in *CBB* model), such as *IBM* [IBM Shopping] and *Sony* [Sony] that provide online recommendation systems for their computer systems.

IBM gives recommendations to customers on computer purchase. The computers are described by the following eight attributes: minimum processor speed, minimum standard disk storage, minimum standard memory, maximum base unit price, operating system, screen size, travel weight, and dockability. *IBM* first recommends notebook computers that are highest priced, median, lowest priced according to the eight features selected by the customers. After that, customers can view more systems or add accessories and upgrades.

IBM recommendation system is mostly a content-based filtering system. It allows customers to select the features they would like to have in their configured system. By matching the items in a wide range of high-quality *IBM* components with customer preferred and/or required contents, the system searches for the computers that best suit their needs.

Sony uses *Active Decisions* [Active Decisions] guided selling technology to engage customers who are shopping for computers. It provides two methods to get highly personalized and completely unbiased product recommendations. One is *Power Search*, which provides recommendations quickly based on the customers' price and

feature preferences. Another is *Get Advice*, which gets recommendations based on how the customers intend to use their Sony Product.

Sony recommendation system is mostly a knowledge-based system. It applies knowledge about customers and computers to generate a recommendation, reasoning about which computer meet the customer's requirements. *Power Search* is aimed at the knowledgeable time-sensitive buyer who wants a quick recommendation that matches price, brand, or feature preferences. To insure that buyers always receive some actionable device, it uses "Fuzzy Recommendations" [DEMOletter, 2001] to identify products that are on the border of a customer's expressed interests, and that would be eliminated by ordinary database queries. *Get Advice* helps buyers new to a product category by asking questions about how they intend to use the product. Unbiased recommendations are extracted from a database of consumer preferences.

In conclusion, agents may be used as mediators in electronic commerce. The role of agents in e-commerce can be classified by their relation to the stages of *CBB* model. The majority of agents in product recommendation systems is developed using content-based, collaborative-based, constraint-based filtering or case-based reasoning. These technologies have their own characteristics as we discuss before respectively. We applied both case-based reasoning and collaborative-based filtering technologies and proposed some novel methods of case-based reasoning to construct our *PCFinder* as we will present in the following chapters.

Technologies Used in PCFinder

In order to provide some service for customers and offer marketing analysis tools to management staff in an online computer store, we have constructed an intelligent product recommendation agent, which works for the configuration of personal computers, based on CBR and collaborative filtering technologies, called *PCFinder*. *PCFinder* makes use of five different technologies: agent technology, Case-Based Reasoning, collaborative filtering, clustering techniques and XML technology. In the following sections, we will introduce each of these technologies in turn.

3.1 Agent Technology

In this section, we explain what an agent is and provide a classification of agents.

3.1.1 Introduction to Agents

“An *agent* is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors” [Russell & Norvig, 1995].

Agents can also be described as “autonomous and computational entities capable of solving problems and effective operation in dynamic and open environments. In

multi-agent systems, agents interact and cooperate with other agents (including both people and software) that may have conflicting aims. Agents are different from objects (in the sense of object-oriented software), because they are autonomous entities capable of making choices over their actions and interactions. Agents cannot be directly invoked like objects, but they can be constructed using object-oriented technology". [Luck *et al.*, 2002]

The environment in *PCFinder* includes two parts. One is the database in which are stored the product information, customer profiles and purchase records used for search and recommendation tasks. The other is the Human Computer Interface (HCI) used to interact with the user. The *PCFinder*'s percepts are the words of an *HTML* (*Hypertext Markup Language*) document acquired from using software sensors that connect through the Internet/Intranet utilizing *HTTP* (*Hypertext Transfer Protocol*). *PCFinder*'s goal is to find an appropriate product matching its search criteria. It acts on both the database and web browsers to update the search result and hopefully to find and deliver the desired end result.

3.1.2 Classification of Agents

There is no easy and straightforward way of grouping agents in to certain classes. Nwana [Nwana, 1996] uses several dimensions to classify the existing software agents [Bradshaw, 1997]:

- The mobility, which means agents are able to move around network. According to the mobility, agents are classified into static agents or mobile agents
- The presence of an internal symbolic reasoning model, as deliberative or reactive
- The attributes that they should exhibit ideally, such as autonomy, cooperation, learning. By using these characteristics, Nwana derives four types of agents: collaborative, collaborative learning, interface, and smart (see Figure 3.1).

- The roles that agents act, as information or Internet
- Hybrid philosophies, the two or more approaches that are combined in a single agent
- The attributes that are considered as secondary attributes, such as versatility, benevolence, veracity, trustworthiness, temporal continuity, ability to fail gracefully, and mentalistic and emotional qualities.

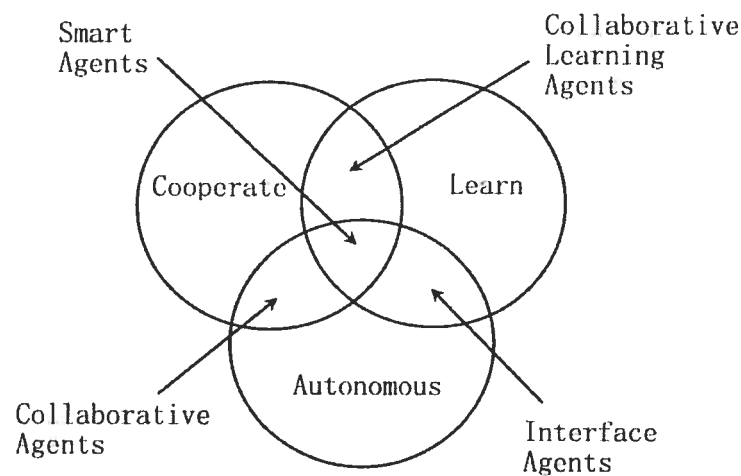


Figure 3.1 Typology based on Nwana's [Nwana, 1996] primary attribute dimension

To combine the previously mentioned properties, the agent typology is basically grouped into seven categories [Nwana, 1996]:

- Collaborative agents
- Interface agents
- Mobile agents
- Information/Internet agents
- Reactive agents
- Hybrid agents
- Intelligent agents

Collaborative agents emphasize the characteristics of autonomy and cooperation with other agents in order to complete tasks for their owners. They do not require the ability to learn. Because they possess the qualities of autonomy, social ability, responsiveness and proactiveness, they are able to act rationally and autonomously in open and time-constrained multi-agent environments [Nwana, 1996].

Interface agents emphasize the characteristics of autonomy and learning in order to complete tasks for their users. The key feature of interface agents is that they are personal assistants that collaborate with the human user in the same working environment. The interaction between the human and the interface agent does not require an agent communication language. Essentially, interface agents help the user to learn how to use a particular software application, such as a spreadsheet or an operating system [Nwana, 1996].

Mobile agents are computational software processes that have the ability to migrate from one computer to another via network. They interact with the other hosts, gather information from them, and finally return to their owner with the gathered information. They have a number of applications, from flight reservations to the management of telecommunication networks. Mobile agents are also autonomous and able to cooperate with other agents. When an agent communicates or cooperates with other agents, it should exchange only the necessary data for the desired purpose, not its entire database. Also, it should not gather the same information twice or transport information back to the location it came from. Furthermore, security issues are important when mobile agents are designed and implemented. For example, as viruses behave in the same way as agents, it is important to ensure that viruses are excluded from agents [Nwana, 1996].

Information agents are tools for gathering the exponentially growing information on the Internet. Their role is to manage, manipulate and collate the information retrieved from distributed sources. This role-based definition is a bit ambiguous, since it can easily fit other agent definitions as well. Hence, this is an issue of debate in defining agents. Information agents can perform different roles depending on the application. For example, *database agents* retrieve information from databases [Nwana, 1996].

Reactive agents are unique in that they do not possess any kind of internal, symbolic model of the environment in which they are embedded; instead they act/respond in a stimulus-response manner to the present state of this environment. Reactive agents are simple so that they interact with other agents in a basic way. Nevertheless, complex patterns of behavior are visible when the interaction between agents is viewed more globally. Because of these properties, reactive agents are often used on raw sensor data that needs to be processed quickly [Nwana, 1996].

Hybrid agents are a combination of the different types of agents just mentioned. Each type of agent has its strengths and its deficiencies. Hybrid agents aim to find the best solution to the problem at hand by maximizing the agent's strengths and minimizing its deficiencies [Nwana, 1996].

“An **intelligent agent** is a computer system that is capable of flexible autonomous action in order to meet its design objectives” [Jennings & Wooldridge, 1998]. Intelligent agents possess some sort of decision-making model that gives them a primitive level of intelligence. This intelligence is usually based on reasoning theory, fuzzy logic, knowledge-based systems, neural networks, or some combination of these types. Generally, an agent is intelligent if it perceives its environment, is capable of reasoning its perceptions, solves problems and determines actions depending on its environment and the tasks provided by its user. Some intelligent agents have the

ability to learn from the users or from other agents. But as building a learning agent is a complex task, most current intelligent agents are not learning agents.

Intelligent agents have many applications in e-commerce. For example, agents can assist the user with online shopping, by searching the web and recommending products that meet constraints specified by the user. An agent can also act as a sales person for a particular vendor by providing product advice or by helping the customer with product troubleshooting [Hermans, 1997].

According to the classification of software agents, our *PCFinder* works as an intelligent agent that is deployed in e-commerce to assist the customer with online shopping. After receiving a request from the customer, *PCFinder* autonomously searches for the personal computer that best satisfies the customer's criteria and then returns this result to the customer. If the customer is not satisfied with this result, *PCFinder* adapts it and returns a more appropriate product to the customer. *PCFinder*'s decision-making model is based on Case-Based Reasoning, which will be discussed in the next section.

3.2 Case-Based Reasoning

In this section, we describe the basics of CBR technology used to realize *PCFinder*. Case-Based Reasoning is the process of solving new problems based on the solutions of similar problems solved in the past. For example, an automobile mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using Case-Based Reasoning. Similarly, a lawyer who advocates a particular outcome to a trial based on legal precedents is using Case-Based Reasoning. CBR is based on the key assumption that if two problems are similar, their solutions are probably also

similar [Bergmann *et al.*, 2002; Aamodt & Plaza, 1994].

3.2.1 The CBR Concept: Using Old Solutions to Solve New Problems

The old problems and their solutions are stored in a database of cases – the case base. When faced with a new problem to solve, the CBR system retrieves the most similar old problem in the case base. The solution to the new problem is derived from the solution to the old problem. This solution can be adapted to satisfy more accurately the requirements of the new problem [Bergmann *et al.*, 2002]. Once the new problem has been solved, its solution is added to the case base for future use. The various steps of the CBR process are illustrated in Figure 3.2.

The CBR process consists of four-steps:

1. Retrieve: Given a new problem, retrieve the cases in memory that are relevant to its solution. A case consists of a problem, its solution, and typically annotations about how the solution was derived.
2. Reuse: Map the solution from the old case to the new problem. This may involve adapting the solution to fit the new problem.
3. Revise: Test the new solution in the real world (or using a simulation) and revise it if necessary.
4. Retain: After the solution has been successfully adapted to the new problem, store the resulting problem/solution pair as a new case in memory.

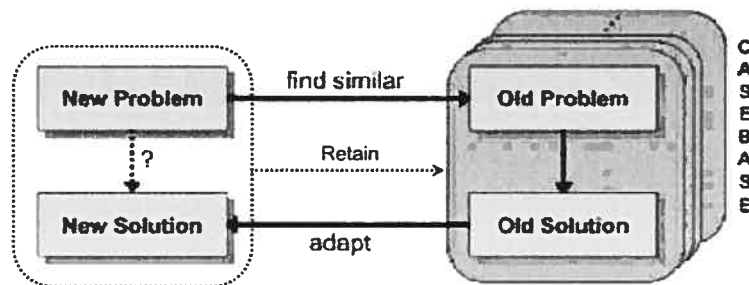


Figure 3.2 Steps of the CBR process [Bergmann et al., 2002]

In CBR systems used for product searches in e-commerce, the cases consist of product descriptions. The problem is a single product query with the various criteria that the product should satisfy. The solution is a specific product that meets these requirements. The solution for configurable products (such as computers, automobiles, complex machines etc.) can be the entire configuration [Bergmann et al., 2002]. A customer's query is regarded as a new problem by the CBR system, and the system searches through the old problems in the case base to find the most similar problem.

3.2.2 Similarity Measures

At the heart of the CBR system is the computation of the *similarity* between the new case entered by the user and the previous cases stored in the case base. Cases are described by qualitative and quantitative parameters called *features* or *attributes*. The CBR algorithm calculates the similarity between cases based on the pairs of values of each feature in the new and old case

The similarity between two cases, also called *global similarity*, is based on the *local similarity*, which is the similarity between each pair of corresponding attributes of these two cases. The local similarity depends on the type of the attribute and the range

its value may take. A similarity value is a real number usually between 0 and 1. For the global similarity, a value of 0 means the old case does not satisfy the new case at all; a value of 1 means the old case matches the new case perfectly. A local similarity value of 0 means the new case does not satisfy the old case for the attribute measured; a value of 1 indicates that the two cases are perfectly matched for that attribute.

In order to find the best solution using the similarity measure, each case can be considered as a fixed length vector of n attributes. These attributes may consist of numerical values or non-numerical values arranged into some kind of order. Figure 3.3 illustrates how similar cases are placed closely together in space. The points represent the cases with n attributes. If two points are close together in space, these two cases will be similar based on the similarity measure. When a new problem appears, the system computes the similarity between this problem and all the old problems in the case base. The system finds all the problems whose similarity is within a certain range and then recommends the solutions of these problems to the user.

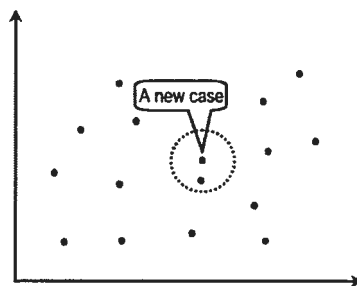


Figure 3.3 Representation of cases with n attributes

Some attributes of a case can be represented using numerical values. For these attributes, a simple approach is to calculate the difference between the query value and the value of the case and normalize the result to the interval $[0,1]$. For example,

the similarity measure can be calculated by applying the formula $1-(d/(d+1))$ to the difference d . [Bergmann *et al.*, 2002].

Some attributes of a case cannot be represented using numerical values. Therefore, the attributes are often represented by a Boolean variable or using a symbolic style. In such cases, a look-up table is used to define the corresponding similarities. In this table, all possible pairs of attribute values are non-numerical values, but that are ordered in a particular order. For example, an operating system used in a personal computer can be put into order by their published dates, such as Windows 98, Windows 2000, Windows XP, etc. Sometimes even more complex attribute types are required, such as taxonomy types or complex objects [Bergmann *et al.*, 2002].

The most frequently used approach of similarity measure between stored cases and the new input case is generally based on matching a weighted sum of features. A usual way for calculating the similarity is to apply a weighted sum of all the local similarity. The similarity σ between a query q and a case c can be formulated as follows:

$$\sigma(q, c) = \sum_{i=1}^n \omega_i \sigma_i(q_i, c_i) \quad \text{where} \quad \sum_{i=1}^n \omega_i = 1 \quad \text{and} \quad \omega_i \geq 0 \quad \text{for all } i$$

This formula states that the similarity σ between a query q and a case c can be calculated using the local similarities σ_i . In this formula, a query q is described by the attributes $q_1 \cdots q_n$, and a case c is described by the attributes $c_1 \cdots c_n$, where attribute values with matching indices correspond to the same attribute. The weights ω_i can be assessed by a domain expert. Also customers can change them in order to express their individual preferences. The details of similarity measure will be discussed in section 5.1.

There are many ways to modify the general definition of a similarity measure as a

weighted sum. Generally both the data model and the type of application should be taken into account in the computation of the similarity measure. The most critical part of the design and implementation of a CBR system is finding a suitable similarity measure [Bergmann *et al.*, 2002]. The similarity measure does not need to be changed frequently; therefore once a good similarity measure has been decided on and applied to the CBR system, maintaining the system is simple.

3.3 Collaborative Filtering

The goal of collaborative filtering is to predict the preferences of a user, referred to as the active user, based on the preferences of a group of users [Pennock & Horvitz, 2000; Sarwar *et al.*, 2000; and Smyth & Cotter, 1999]. The system makes a correlation between the active user and other users whose profiles are similar according to a similarity metric. Based on this correlation, it then finds items that the group has in common, which are missing from the active user profile [Hayes & Cunningham, 2000].

For example, given the active user's ratings of several movies and a database of other users' ratings, the system can predict how the active user would rate unseen movies. The key idea is that the active user will prefer the items that like-minded people preferred, or even that dissimilar people liked less.

The effectiveness of any collaborative filtering algorithm relies on the underlying assumption that human preferences are correlated - if they were not, then informed prediction would not be possible [Pennock & Horvitz, 2000].

Sarwar *et al.* [2000] divide the process of generating collaborative-based

recommendations into three sub-tasks: *representation*, *neighborhood formation*, and *recommendation generation*.

- Representation

This task addresses the representation used to model the products that have already been purchased by a customer.

In a typical collaborative-based product recommendation system, the input data is a collection of purchase histories of n customers for m products, which is usually represented as an $m * n$ customer-product matrix.

- Neighborhood formation

This task focuses on how to identify the other neighboring customers.

The most important step in collaborative-based product recommendation systems is computing the similarity between customers as it is used to form a proximity-based neighborhood between a target customer and a number of like-minded customers.

The proximity between two customers is usually computed using the correlation or the cosine measure.

1. Correlation

In this case, the similarity between two users a and b is measured by computing the Pearson Correlation Coefficient $corr_{ab}$ given by

$$corr_{ab} = \frac{\sum_i (r_{ai} - \bar{r}_a)(r_{bi} - \bar{r}_b)}{\sqrt{\sum_i (r_{ai} - \bar{r}_a)^2 \sum_i (r_{bi} - \bar{r}_b)^2}}$$

Where r_{ai} (or r_{bi}) is one if user a (or b) has purchased the i^{th} product, and zero,

otherwise. \bar{r}_a (or \bar{r}_b) denotes the mean of $\sum r_{ai}$ (or $\sum r_{bi}$).

2. Cosine

In this case, the similarity between two users a and b , who are thought of as two vectors, is measured by computing the cosine of the angle between the two vectors given by

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Where “.” denotes the dot-product of the two vectors. $|\vec{a}|$ (or $|\vec{b}|$) is the distance of vector \vec{a} (or \vec{b}).

- Recommendation generation

The final step of a collaborative-based product recommendation system is to find the *top-N* recommended products from the neighborhood of customers.

3.4 Clustering Techniques

In this section, we provide a brief overview of clustering and cluster analysis. Then we introduce a categorization of major clustering methods.

3.4.1 Clustering

Clustering is the method by which similar records are grouped together. Often clustering is used to provide the end user with a high-level view of what is going on in the database. The term clustering is used sometimes to denote segmentation, which most marketing people will tell you is useful for coming up with a birds-eye view of

the business market [Berson *et al.*, 1999].

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group in many applications [Han & Kamber, 2001].

Cluster analysis has been widely used in numerous applications, including pattern recognition, data analysis, image processing, and market research. Using clustering, one can identify dense and sparse regions and, therefore, discover overall distribution patterns and interesting correlations among data attributes [Han & Kamber, 2001].

3.4.2 A Categorization of Major Clustering Methods

There exist a large number of clustering algorithms in the literature. The clustering algorithm is chosen based on the type of data and on the particular application. If cluster analysis is used as a descriptive or exploratory tool, several algorithms can be tried on the same data to see what information the data may disclose.

Clustering methods can be broadly divided into partitioning method, hierarchical method, density-based method, grid-based method and model-based method [Han & Kamber, 2001].

The **partitioning method** constructs an initial set of k partitions, and then improves the partitioning by moving objects from one group to another using an iterative relocation technique. The most generally used partitioning methods are *k-means*,

k-medoids, and their variations [Han & Kamber, 2001].

The **hierarchical method** constructs a hierarchical decomposition of the set of data objects. The method can be categorized as either agglomerative (bottom-up) or divisive (top-down) based on the form of hierarchical decomposition. Recent studies have emphasized the combination of hierarchical agglomeration and iterative relocation methods [Han & Kamber, 2001].

The **density-based method** groups objects based on the notion of density. To discover clusters with arbitrary shape, density-based clustering methods have been developed. These methods generally regard clusters as regions of high object density in the data space separated by regions of low density [Han & Kamber, 2001].

The **grid-based method** first divides the object space into a finite number of cells so that a grid structure is constructed, and then it clusters objects on this grid structure. The main advantage of this approach is its fast processing time, which is usually independent of the number of data objects and depends only on the number of cells in each dimension in the quantized space [Han & Kamber, 2001].

The **model-based method** assumes a model for each of the clusters and then finds the best suitable data to this model. Typical model-based methods include *statistical approaches* and *neural network approaches* [Han & Kamber, 2001].

3.5 XML Technology

XML (eXtensible Markup Language) is a subset of the *Standard Generalized Markup Language (SGML)* defined by *W3C (World Wide Web Consortium)*, which provides a

way to create extensible formats for describing structured data in electronic documents and to express rules about those data, so that it becomes easier to interchange structured documents over the Internet [XML 1.0; Bryan, 1997].

In order to visualize what XML documents look like, a simple XML document is shown as follows:

```
<?xml version="1.0"?>
<Library>
  <Book>
    <Title>Decision Support Systems and Intelligent Systems</Title>
    <Author>Efraim Turban & Jay E. Aronson</Author>
  </Book>
  <Book>
    <Title>Developing Intelligent Agents for Distributed Systems</Title>
    <Author>Michael Knapik & Jay Johnson</Author>
  </Book>
  <Book>
    <Title>Constructing Intelligent Agents with Java</Title>
    <Author>Joseph P. Bigus & Jennifer Bigus</Artist>
  </Book>
</Library>
```

The document demonstrates a number of the rules found in an XML document. The first line in the above sample is the XML declaration, which defines the XML version of the document. In this case the document conforms to the 1.0 specification of XML. Documents do not have to include this element, but normally it should always be included. All XML documents must have one enclosing element (the version information does not count as an enclosing element). The Library element wraps the entire document above. It is the first element of the document (the root element). It contains three sub-elements: three books. Each sub-element also has two sub-elements: title and author. The last line defines the end of the root element. Not one word in the above XML document is an XML keyword. So the most important for

a freewheeling XML author is keeping the spelling in the tag names correct and make sure that each individual begin tag has an end tag.

In this chapter, we introduced five different technologies used in *PCFinder*, such as agent technology, Case-Based Reasoning, collaborative filtering, clustering techniques and XML technology. In the following chapter, we will present the architecture of *PCFinder*.

Architecture of PCFinder

Our *PCFinder* prototype uses a typical three-tier architecture, which is more flexible and scalable than the traditional two-tier client/server architecture. The first tier provides a way to present data to the user. The middle-tier is in charge of retrieving the data from the data sources and provides a well-defined interface for the first tier to access the data. The third tier contains the data sources.

4.1 Two-tier versus Three-tier Architectures

In two-tier client/server architecture, applications are closely tied to vendor-specific software. Typically, two-tier applications access database services or transaction services directly from the client. Such applications are sometimes called fat clients because the application logic resides on the client, making the clients large and complex [IBM Education].

Three-tier client/server architectures employ an intermediary, or middle-tier, application server, which operates between client applications and the back-end databases. The middle-tier houses the business logic of the system and coordinates the interaction of the presentation on the client with the back-end databases [IBM Education].

There are two fundamental motivations for using a three-tier architecture over a

two-tier model [IBM Education]:

- Improved scalability, availability, and performance
- Improved flexibility and extensibility of business systems

Two-tier systems perform well by leveraging the processing power of the client, but the dedicated nature of many clients to a single back-end resource, like a database, produces a bottleneck that inhibits scalability, availability, and performance as client populations grow large. Three-tier systems attempt to mitigate this bottleneck by managing back-end resources more effectively. This is accomplished through resource management techniques like pooling and clustering of middle-tier servers. Pooling makes three-tier systems more effective by allowing many clients to share scarce resources like database connections, which reduces the workload on back-end servers. Clustering makes three-tier systems more available and scalable because multiple servers and resources can support fail-over and balance the loads of a growing client population [IBM Education].

Three-tier systems are more flexible and extensible than their two-tier counterparts because the business logic and services, such as security and transactions, reside on the middle-tier and are largely hidden from the client applications. If properly implemented, as is the case with Enterprise JavaBeans, services are applied automatically to client requests and are therefore invisible. Because services are not visible to the client, changes to services are also invisible. Changes and enhancements to business logic on the middle-tier can also be hidden from client applications if implemented correctly [IBM Education].

4.2 Architecture of PCFinder

The three-tier architecture of *PCFinder* is shown as Figure 4.1.

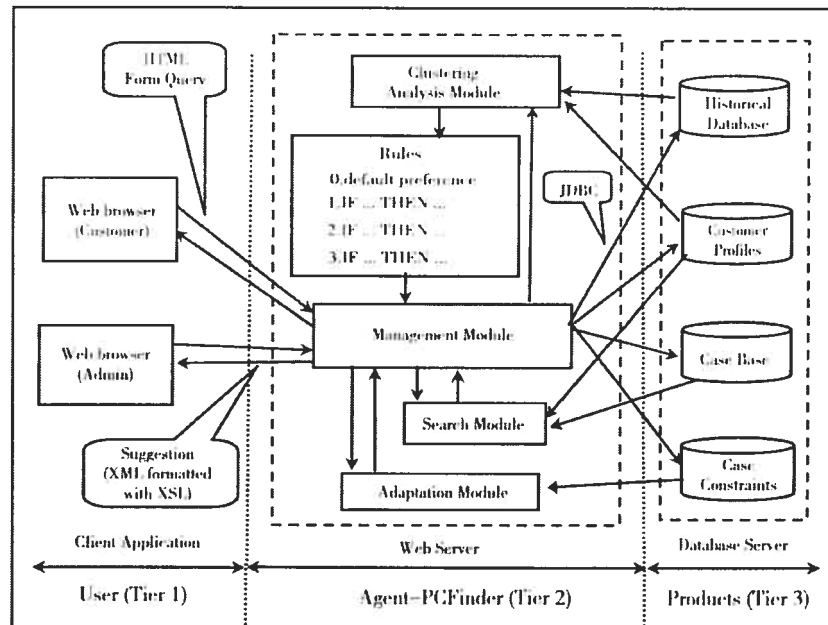


Figure 4.1 Three-tier architecture of *PCFinder*

The first tier of our application includes personal computers with browsers. In this thesis project, we provide two kinds of user interfaces based on the I.E. 5 web browser. One is for general or registered users; the other is for administrators and management staff.

The intelligent agent, *PCFinder*, runs in the middle-tier. It includes four modules: a management module, a search module, an adaptation module and a clustering analysis module. It also includes generative rules. The management module is in charge of interacting with the first tier, managing the other modules and maintaining the databases in the third tier (see Figure 4.2). The search module helps the customer to find the most appropriate product from the product base (case base) according to the

customer's query and long-term constraints (see Figure 4.3). The adaptation module modifies products to better fulfill the customer's needs when the customer thinks it is necessary to do so (see Figure 4.4). The clustering analysis module analyzes and groups the information of customer profiles database and historical database, and generates rules so that management module can adjust some initial values, such as initial weight of the attributes (see Figure 4.5).

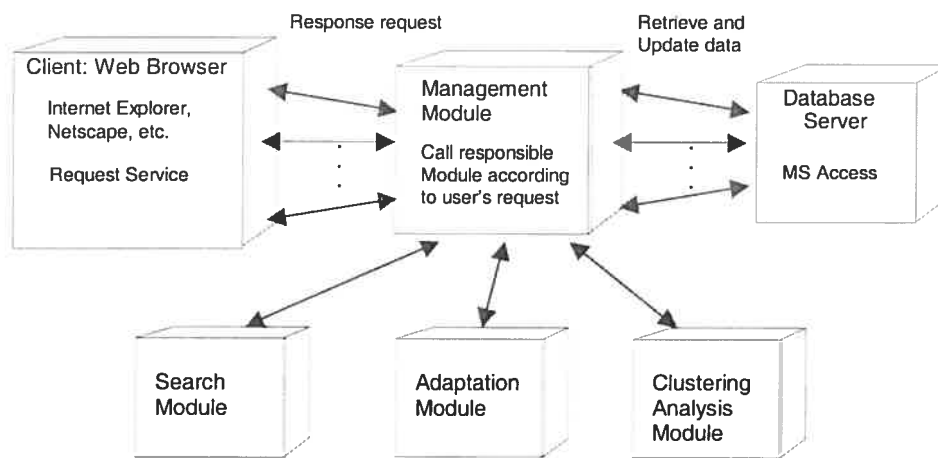


Figure 4.2 Management module

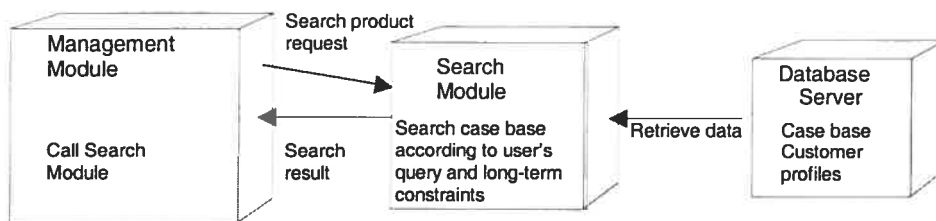


Figure 4.3 Search module

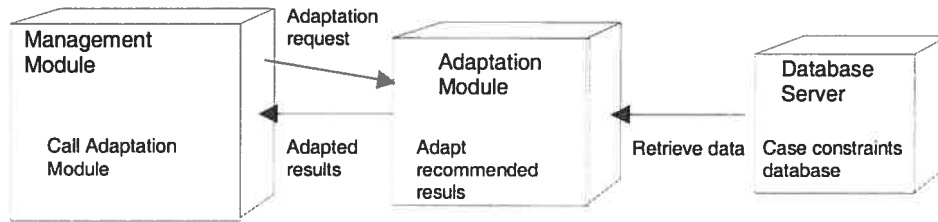


Figure 4.4 Adaptation module

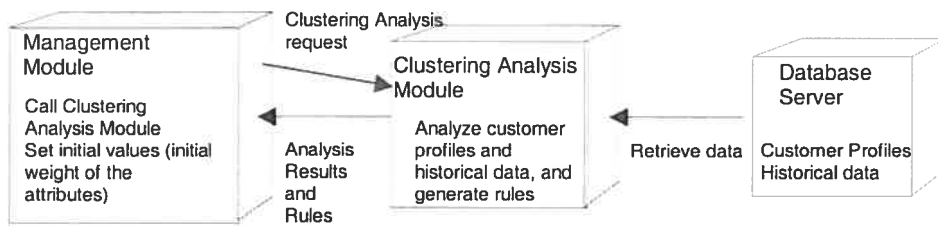


Figure 4.5 Clustering analysis module

The third tier contains the database. It includes a case base that contains product data, a case constraint database that contains the adaptation criteria, and a customer profile database that contains customers' background and long-term constraints, and a historical database that contains the historical purchasing records of each customer.

In our *PCFinder*, the first tier sends HTML form queries to the middle-tier by encoding commands and arguments in HTTP requests. The middle-tier retrieves the data from the case base in the third tier or maintains the data in the third tier by JDBC (*Java Database Connectivity*). Then it responds with XML documents formatted with XSL (*eXtensible Stylesheet Language*). Using XML instead of HTML means that we can have dynamic content without sacrificing usability or interoperability.

Figure 4.6 depicts the transformation process from XML to HTML [Allamaraju *et al.*, 2001]. An XML document provides an input source to an XSLT processor. An XSLT processor is a software component that is able to read an input source, apply rules to

the input source in the form of an XSL Style Sheet, and produce an output document that conforms to the rules in application. In *PCFinder*, the XSLT processor applies rules that generate an HTML document from the input XML source.

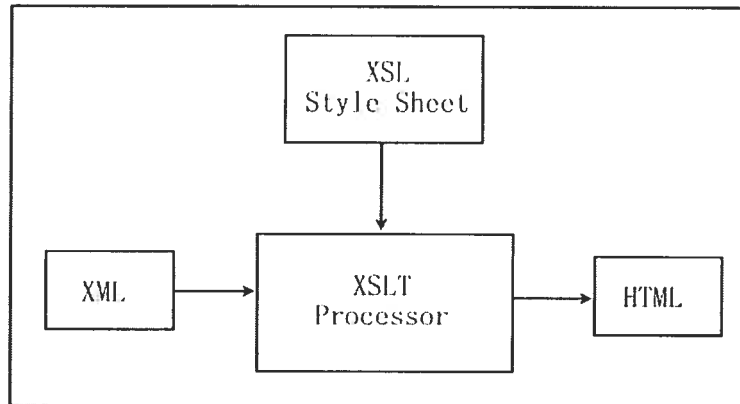


Figure 4.6 The transformation process from XML to HTML

In this chapter, we first explain why three-tier architecture is more flexible and scalable than the traditional two-tier client/server architecture. Then, we introduce the three-tier architecture of our *PCFinder*. In the following chapter, we will introduce the Methodologies and Algorithms used in our *PCFinder*.

Methodologies and Algorithms

In the following sections, we first present three novel methods: order-based similarity measure, weight modification and adaptation. Then we introduce profile and collaborative filtering, clustering analysis and two other concepts used in our *PCFinder*, namely long-term and short-term profiles.

5.1 Dynamic Order-Based Similarity Measure

Local similarity measures largely depend on the application domain, but they all serve the same use: to return an estimation between 0 and 1 and to indicate the similarity between a particular attribute of a case and its equivalent in the request [Aïmeur & Vézeau, 2000].

Here, we present a method of *Order-Based Similarity Measure*. This method is based on *Order-Based Retrieval* [Bridge, 2001]. The customer supplies a variety of information (preferred values, values to be avoided, maximum values and minimum values, for example) and we construct an ordering relation from this information. Then we can use this ordering to calculate the local similarity.

In order to demonstrate *Order-Base Retrieval*, Bridge [2001] draws a distinction between *ordered types* and *unordered types* information retrieved from users. An ordered type is one that has a non-trivial partial order of its values that may be useful in product recommendation. For example, the attribute of *price* is an ordered type,

since its type is numeric, the values are ordered by the usual ordering of the numbers. An unordered type, by contrast, is one whose values have no non-trivial ordering that is relevant to product recommendation. Instead, they assumed that for unordered types there exists a relevant similarity measure on the values. For example, for a holiday case base, the attribute of *transport* might have an unordered type $\{plane, train, car, coach\}$, which type is non-numeric [Bridge, 2001]. For our Order-Based Similarity Measure, we assume that information on both *ordered types* and *unordered types* can be put into order by domain experts.

The advantage of the Order-Based Similarity Measure is that it is easy to maintain the similarity attribute-value pair table. When the order of a specific attribute of some cases has been decided, it is only a simple calculation away from getting the similarity between the attribute-values. We can get the similarity attribute-value dynamically rather than from pre-initialization. For example, when a new attribute of some cases is added in the case base, we do not need to update the similarity attribute-value pair table saved in XML documents or other databases.

Consider the set $V = \{a_1, a_2, \dots, a_n\}$ of values for a specific attribute of some cases, which excludes the values that are not considered by users. Assume that $a_{i-1} < a_i$ for each i , $1 < i \leq n$, where n is the maximum number of possible values within the range acceptable to the customer. We say that i represents the *serial number* of value a_i in V . Let q be the customer's "ideal" value for this attribute. If $q \in V$, let m be its serial number, i.e. $q = a_m$. Assuming for simplicity that $q \in V$ whenever $a_1 \leq q \leq a_n$, we define the *local similarity measure* as follows:

$$S(q, a_i) = \begin{cases} 1 - |i - m| / \max[(n - m + 1), m] & \text{if } q \in V \\ 1 - (n - i + 1) / (n + 1) & q > a_n \\ 1 - i / (n + 1) & q < a_1 \end{cases}$$

For example, we wish to find a computer whose *processor speed* is 800MHz, but we are willing to consider speeds between 700MHz and 1000MHz. The possible values of *processor speed* and the corresponding local similarity measures are shown in Table 5.1. In this case, speeds above 1000MHz have zero similarity, so that $n=7$, $V=\{700, 750, 800, 850, 900, 950, 1000\}$, $q=800$, $m=3$, and therefore $S(q, a_i) = 1 - |i - 3| / 5$.

Table 5.1 Local similarity measures for the "processor speed" attribute

Serial Number (<i>i</i>)	1	2	3	4	5	6	7	(Not consider)	
Processor Speed (MHz)	700	750	800	850	900	950	1000	1200	1400
Similarity	0.6	0.8	1.0	0.8	0.6	0.4	0.2	0	0

We will now look at an example using unordered type. In the case of the *brand* attribute, we assume that 1) we have a similarity measure on different manufacturers, and 2) some experts, such as marketing analysts, etc., will provide an order of the brands. For example, the order of *brand* might be $V=\{\text{Acer, Compaq, HP, IBM, Panasonic, Sony, Toshiba}\}$. If we want to find a computer whose *brand* is Sony, the possible values of *brand* and the corresponding local similarity measures are shown in Table 5.2. In this case, $n=7$, $q=\text{Sony}$, $m=6$, and therefore $S(q, a_i) = 1 - |i - 6| / 6$.

Table 5.2 Local similarity measures for the "brand" attribute

Serial Number (<i>i</i>)	1	2	3	4	5	6	7
Brand	Acer	Compaq	HP	IBM	Panasonic	Sony	Toshiba
Similarity	0.17	0.33	0.5	0.67	0.83	1.0	0.83

5.2 Weight Modification

The Global Similarity Measure is computed typically by taking a weighted average of the local similarity measures. The *weights* ω_i provide some indication of the relative importance of the different attributes. These are the quantities that we want to modify when the user is not satisfied with some of the proposed specific attributes. Following a critique from the user, the main task of the system consists of computing how much change we should make to the weight(s) of certain attribute(s) that the user considers to be the most important to him. According to this, we should compute the similarity with the weight, Sim_i , of each attribute using this formula:

$$Sim_i = \omega_i S_i(q_i, c_i)$$

Where S_i is the similarity of each attribute in certain case, q_i and c_i , are the attribute of query and the attribute of the case. Once the similarity of each parameter i is computed, we cover one by one each parameter whose similarity is higher than the criticized parameter, Sim_* , which is considered by the user to be the most important to him. For all those parameters, we reduce their weight as:

$$\omega_i \leftarrow \omega_i - (Sim_i - Sim_*) \quad \forall i \quad Sim_i > Sim_*$$

Finally, if there were at least one parameter whose similarity is higher than Sim_* , the weight of the criticized parameter is increased by an amount equivalent to the sum of all reductions to the weights of the parameters that lost importance, so that the sum of all weights stays equal to 1:

$$\omega_i \leftarrow \omega_i + \sum (Sim_i - Sim_*) \quad \forall i \quad Sim_i > Sim_*$$

As an example, imagine that the system is in the state illustrated in Table 5.3 and the customer wishes to criticize the brand “IBM”. Table 5.4 represents the system state after this critique.

Table 5.3 The system state before the “brand” critique

Item description	Query	Result	Similarity	Weight
Processor speed	1600 MHz	1600 MHz	1.0	0.125
Memory	128 M	256 M	0.75	0.125
Hard disk drive	30 G	20 G	0.67	0.125
Display	12.1"XGA TFT	14.1"XGA TFT	0.83	0.125
Multimedia	CDROW	DVD	0.5	0.125
Operating System	Win 2000	Win 2000	1.0	0.125
Brand	IBM	Panasonic	0.75	0.125
Price	2001-2500\$	2149.5\$	1.0	0.125

Table 5.4 The system state after the “brand” critique

Item description	Query	Result	Similarity	Weight
Processor speed	1600 MHz	1600 MHz	1.0	0.093
Memory	128 M	256 M	0.75	0.125
Hard disk drive	30 G	30 G	1.0	0.125
Display	12.1"XGA TFT	15"XGA TFT	0.67	0.115
Multimedia	CDROW	CD-RW	0.75	0.125
Operating System	Win 2000	Win XP Pro	0.33	0.093
Brand	IBM	IBM	1.0	0.228
Price	2001-2500\$	2313.95\$	1.0	0.093

Although we provide a method to modify the weights automatically, the customers can change the weights by themselves in order to express their individual preferences. By automatically modifying the weights as the user criticizes the products, the proposed heuristic delivers an increasingly faithful representation of the user's requirements. Note that we assume that the client always criticizes the attribute(s) that he does not like. In this case, the method offers a way to accelerate the convergence toward a recommendation that fulfills the needs of the customer.

5.3 Adaptation

Some CBR systems not only support the retrieval of old solutions, but also adapt these solutions to a new problem, thus creating new solutions that differ from the old ones. Once a matching case is retrieved, the CBR system adapts the solution stored in the retrieved case to the needs of the current case. *Adaptation* looks for prominent differences between the retrieved case and the current case and then applies formulae or rules that take these differences into account when suggesting a solution [Watson & Marir, 1994].

Janet Kolodner gives a definition of *adaptation* as follows [Kolodner, 1993]:

INPUTS:

- A problem description,
- A not-quite-right solution, and
- A problem description that goes with the solution (optional).

OUTPUT:

- A solution that best fits the problem description.

METHOD:

- Adjust the not-quite-right solution to make an appropriate solution for the described problem.

To implement the adaptation step, the knowledge needed to perform the solution modification must be represented in a suitable form in the CBR system. Depending on the application domain, the adaptation process can be more or less complicated.

Adaptation in CBR-based product recommendation systems is especially important for complex, configurable products. If there are many product variants (such as for configurable personal computers or other technical equipment, travel packages etc.), it is not feasible to develop a case base that explicitly stores each possible product configuration. Instead, the initial product is chosen from a limited set of typical base products and then this product is adapted to better suit the customer's requirements.

Our *PCFinder* works for the configuration of personal computers. It is a highly structured domain [Stahl & Bergmann, 2000] in which it is possible to subdivide the problems that arise and their related solutions into more or less independent sub-problems and related sub-solutions. This means that the complete problems, which are also called complex problems, can only be solved if all sub-problems included in each problem are solved. Generally, sub-problems are interdependent; therefore the solution of one often depends on the solutions to the others. It is therefore inevitable to take into account this independence when solving individual sub-problems and when combining them to obtain the final solution.

Because of the interdependence between individual sub-problems, consumers must be reasonable in their product requests if they want to obtain the desired solutions. If the customer makes an unreasonable demand in the definition of a product family, the adaptation module cannot deliver a solution and it may tell the customer that his query is not reasonable. For example, if the customer wants a computer which *processor*

speed is 2000MHz and *price* is lower than \$1000, the adaptation module will tell him that it is impossible to satisfy both of them at the same time.

There are four major steps in our adaptation module:

1. Identifying what needs to be adapted;
2. Identifying what part of the faulty solution should be changed in order to carry out the adaptation;
3. Identifying applicable adaptation methods and/or heuristics; and
4. Selecting an adaptation strategy and implementing it.

Before we present the method used in the adaptation, it is useful to provide a few definitions concerning the attributes (also called sub-problems or sub-solutions) in the query and the case base. We divide the attributes in the query and case base into three types. The first type consists of *independent attributes*, whose values can be changed without being restricted by other attributes, such as hard disk drive and computer model. The second type consists of *dependent attributes*, whose values depend on the *independent attributes* when they need to be changed, such as main memory and CPU. The third type consists of *related attributes*, whose values change automatically when the other attributes change, such as price. Some independent attributes have no dependent attributes associated with them. The relationship between these types of attributes is illustrated in Figure 5.1.

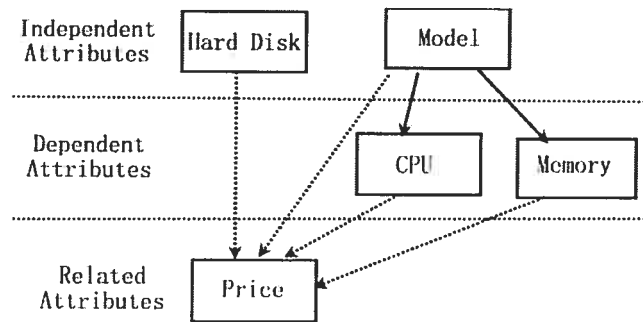


Figure 5.1 Relationship between independent, dependent and related attributes

Now, we introduce some scenarios about the method of our adaptation.

Adaptation of Independent Attributes:

1. Search the capability of independent attribute in the constraint database;
2. If the capability is not less than the customer's requirements, adapt the similar value of the independent attribute to the customer's required value; and adapt the value of the relevant related attributes;
3. Otherwise, return "cannot explicitly be configured".

For example, if the customer is satisfied with the *main memory*, *CPU* and some other attributes but not with *model*, the adaptation module of our agent will search the *model* in the constraint database and checks whether it has the capability to install the attributes with which the customer is satisfied. If so, the adaptation module of our agent creates a new case that satisfies the customer's demand. If not, it recommends the most suitable case.

Adaptation of Dependent Attributes:

1. Search the capability of independent attribute on which the dependent attribute depends in the constraint database;
2. If the capability is not less than the customer's requirements (if there is no conflict with the independent attribute), adapt the similar value of the dependent attribute

- to the customer's required value; and adapt the value of the relevant related attribute;
3. Otherwise, return "cannot explicitly be configured".

For example, if the customer is satisfied with the *model*, *CPU* and some other attributes but not with *main memory*, the adaptation module of our agent will search the *model* in the constraint database to check if this kind of *model* has the capability to install the desired value of *main memory*. If yes, the adaptation module of our agent will create a new case that satisfies the customer's demand. Otherwise, it will tell the consumer that the most similar case, which the search module has found, is the most suitable for him.

Adapting both the *Independent Attributes* and the *Dependent Attributes*:

1. Adapt *independent attribute*; and
2. Adapt *dependent attribute*.

We do not recommend adapting *related attributes* because this type of attributes is related to the other attributes. Its value is determined by the other attributes.

In our *PCFinder*, if a customer is still unsatisfied with the result after modification of the weights, he can turn to the adaptation module of our agent for help. The adaptation module of our agent can automatically estimate if the user's request is reasonable. If yes, the agent applies the adaptation method and creates a new solution that satisfies the user's requirement based on the old solution. If not, the agent informs the user that his requirement cannot be explicitly configured. The framework of the adaptation module is shown in Figure 5.2.

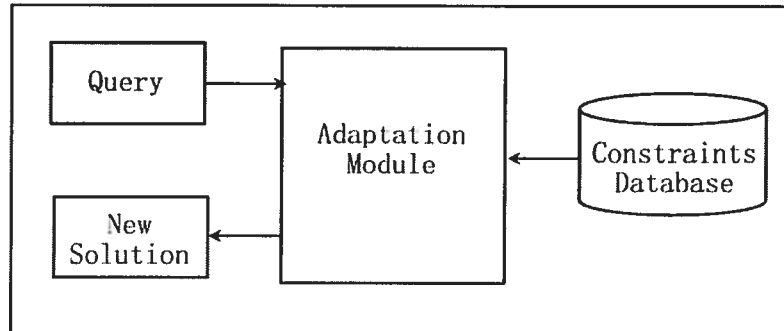


Figure 5.2 The framework of the adaptation module

For example, suppose the customer wants to buy a notebook computer with a *processor speed* of 700MHz and a *RAM* capacity of 512M. The most appropriate notebook in the case base is the model *Versa Sxi* with a *processor speed* of 700MHz and a *RAM* capacity of 128M. If desired so by the customer, the adaptation module of the agent can modify the *RAM* capacity and create a new case of computer. Before modifying the *RAM* capacity, the agent must check if the maximum capacity of the *RAM* in this *model* is equal or larger than 512M. If so, it can modify this attribute and create a new case. Otherwise, it must tell the customer that this *model* cannot install a *RAM* of 512M.

This could be programmed in the following way:

```

if (query.RAM != similar.RAM) {
  if (query.RAM <= similar.Max_RAM) {
    solution.RAM=query.RAM
  }
  else {
    output("This model cannot install so much memory that you preferred")
  }
}

```

5.4 Profile and Collaborative Filtering

In order to customize the system to meet the customer's needs, *PCFinder* provides a collaborative recommendation feature based on the customer's profile. In our system, we differentiate between long-term and short-term profiles. It is important to select the appropriate profile attributes that are relevant to our product recommendation. We only take some attributes as examples for our *PCFinder* and leave the selection of appropriate profile attributes to marketing analysts or domain experts.

5.4.1 Long-term and Short-term Profiles

The customer's shopping habits, such as preferences or constraints, usually last a time. We collect them in the *long-term profile*. This provides very important and useful information to us when we try to get the customer's requirements. Each user is associated with a single profile, and each profile contains user information such as personal identification and selection information. Personal identification information contains various demographic data such as the user's name, age, gender, home address, occupation, credit-card details etc. Selection information is the most important type of information in the profile. It contains a list of products that the user has expressed an interest in or has explicitly ignored in the past.

When a consumer configures his computer, he can provide to our agent a preferred value, a forbidden value, as well as maximum and minimum values for each attribute. For example, a customer might prefer a notebook computer that has a processor speed of 900MHz but not 1000MHz; and he might expect a price range between \$1500 and \$2000. But such preferences are temporary, and therefore we collect them in the *short-term profile* [Aïmeur & Vézeau, 2000]. If the user decides that he cannot afford this particular notebook computer, he may change his requirements. Therefore we

think the *short-term profile* is a valid method to use in the CBR cycle. Therefore we consider that among the various critiques formulated by a customer when searching for the ideal product, the short-term requirement is a good source of information to construct the profile automatically.

5.4.2 Profiling and Collaborative Recommendation

A user profile stores the background of an individual user on the server as a profile database. The key issue in recommendation is the ability to combine a target user with a group of other users that have a profile similar to the target user. The user information contained in each profile can be separated into three basic categories [Cunningham *et al.*, 2001]:

1. **Personal Information:** It includes various personal details such as the user's name, gender, home address, telephone number, email address etc.
2. **Long-term Constraints:** It contains user information relating to the user's preferences, such as what brand or operating system the user will not buy, or the user's preferred price range etc.
3. **Background Information:** This is the most important type of profile information from the collaborative recommendation viewpoint. It contains the user's occupation, age group and salary, the intended use of the computer, the location of computer use etc.

The collaborative recommendation service in *PCFinder* recommends products to target users based on their user profile data; in this sense, the recommendations are personalized for the user in question. The operation of the collaborative recommendation process is described in detail in [Smyth & Cotter, 1999] and [Hayes & Cunningham, 2000]. The key issue in recommendation systems is the ability to combine a target user with a group of other users with similar profiles [Cunningham

et al., 2001]. Profile similarity is a measure of the correlation between the selection lists of the two user profiles; users with a high degree of similarity tend to grade the same products in the same way. A group of users similar to the target user form a virtual community for the target, and recommendations to the target are drawn from the profiles of these community members. The result is a list of recommendable products ranked for example according to the frequency of the product in community member profiles.

The three steps of collaborative recommendation are described as follows:

1. Identify the group in which the given target user belongs.
2. Produce a list of recommendable products or attributes. These products or attributes are ranked according to their appearance in the past purchasing history.
3. Recommend the top n recommendable products or attributes.

The final output of the collaborative recommendation service is a list of products, and ultimately these can be recommended directly to users or combined with the case-based reasoning recommendation. By grouping customers' demographic profiles (such as profession, gender, salary, etc.), the collaborative recommendation service is responsible for identifying virtual communities (as groups of user IDs) within the *PCFinder* user population and for associating individual users with the appropriate community.

5.5 Clustering Analysis

Although there exist many kinds of clustering techniques, we use a simple one to illustrate how it supports product recommendation.

More specifically, *PCFinder* groups customers according to the profiles of their background, such as the main intended use of computer, the location of computer use, the user's occupation, age group and salary range. These user attributes are called *external attributes*. When a new customer signs up, *PCFinder* provides suggestions about the computer configurations according to an analysis of the purchasing history of all previous customers who have a similar profile. These computer configurations are called *internal attributes*. Thus *PCFinder* gets internal attributes by analyzing external attributes using clustering techniques. But if it is the first ever purchase, *PCFinder* will return a warning message advising that there is no similar profile from the history.

Some cases and solutions about clustering analysis are described as follows:

Case 1:

PCFinder suggests the most popular used internal attribute by clustering their external attributes.

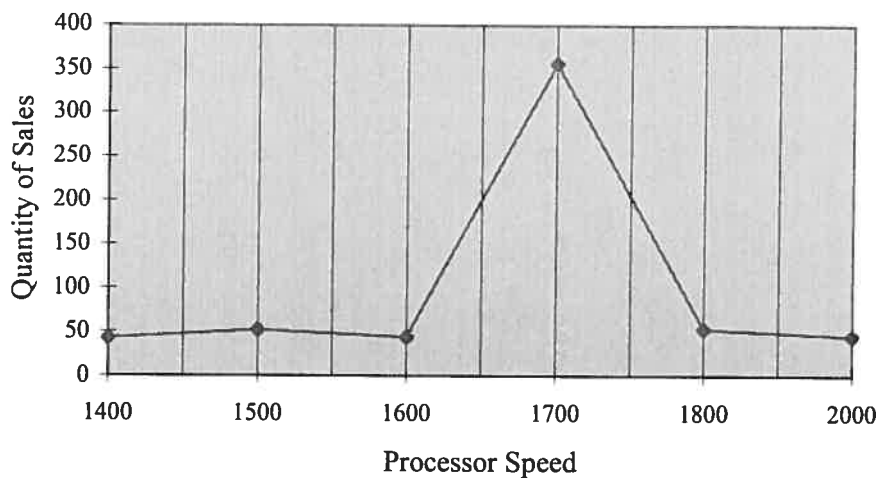


Figure 5.3 Quantity of sales based on purchase records versus processor speed

For example, *PCFinder* groups customers who use their computer for playing games and discovers that most of them buy computers whose processor speed of 1700MHz. The relation between processor speed and the quantity of sales based on historical purchase records is shown in Figure 5.3. Hence, such computers are recommended to new customers who intend to use their computer for playing games.

Case 2:

PCFinder suggests the most popular used internal attribute and gives this internal attribute a higher weight than the others by clustering their external attributes.

For example, *PCFinder* groups customers who work as university professors and discovers that most of them have bought either Compaq or SONY. The relation between brand and the quantity of sales based on historical purchase records is shown in Figure 5.4. Hence, these two brands are recommended to new customers who are university professors, and the initial weight of “brand” is increased.

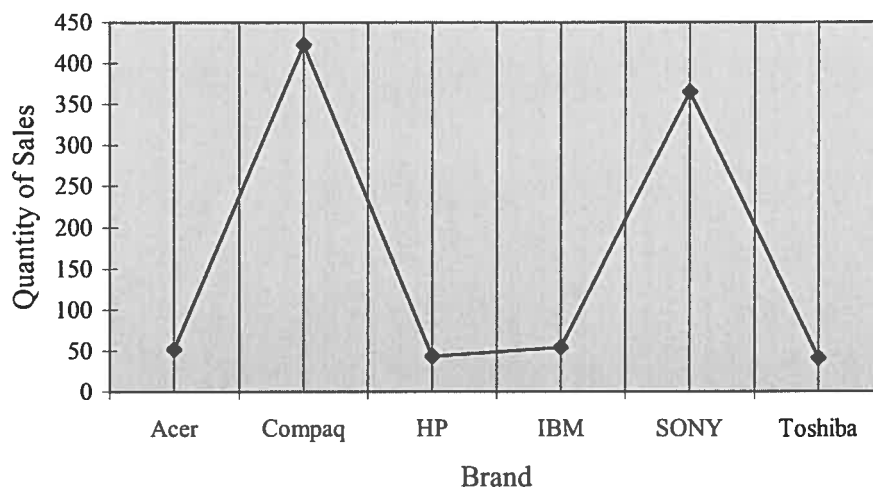


Figure 5.4 Quantity of sales based on purchase records versus brand

Case 3:

This case is more complicated than the first two. In case 1 and 2, clusters take account of a single attribute. But in the case base, there are usually several attributes. Three attributes in a case can be represented as in Figure 5.5.

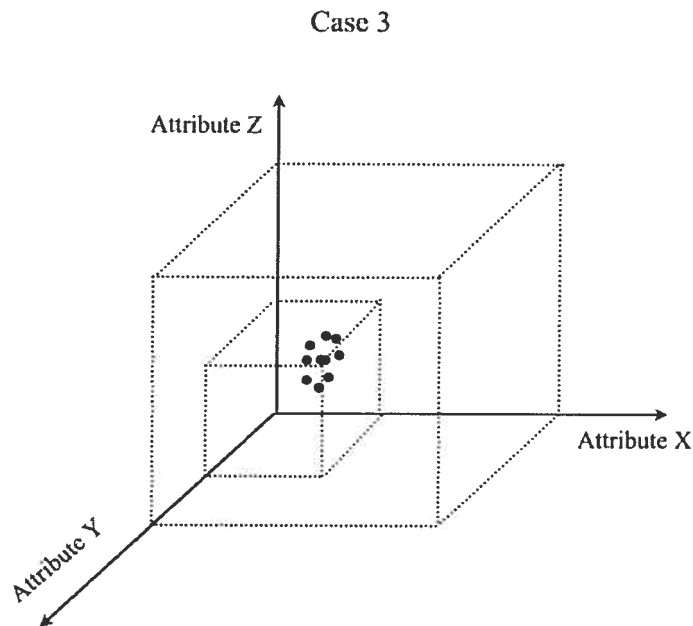


Figure 5.5 The representation of three attributes in a case

In this case, *PCFinder* maps groups of related user profiles to groups of related computer configurations. For each new customer, *PCFinder* finds out to which group he belongs when he fills out his profile form. By analyzing cluster correlations between external and internal attributes, *PCFinder* recommends the configuration of the cluster that correlates best with the customer's external attributes (See Figure 5.6).

For example, *PCFinder* could group customers who are university students, who use computers to play games and whose age is between 21 and 30. Using techniques from Cases 1 and 2 above, *PCFinder* discovers that most of these customers bought SONY

or Compaq computers whose processor speed is 1700MHz, operating under the Home Edition of Windows XP. Therefore, these configurations are recommended to new customers who have the same profile as this group.

Correlation between external attributes and internal attributes

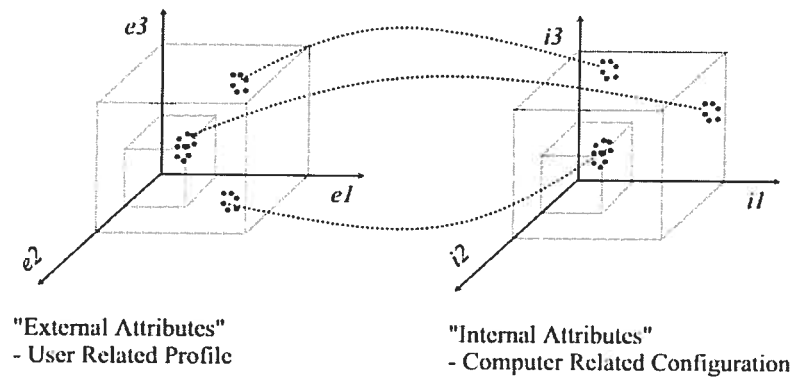


Figure 5.6 The correlation between external and internal attributes

In this chapter, we propose some novel methods based on CBR for an e-commerce application, which includes Order-Based Similarity Measure, weight modification and adaptation methods. Collaborative filtering and the simple clustering technique are also introduced. In the next chapter, we start to detail the implementation of our system based on the methods described in this chapter.

Implementation of PCFinder

In order to illustrate the architecture and methodology of our product recommendation system, we constructed an intelligent agent – *PCFinder* – that runs on an online notebook computer store to provide suggestions to customers as well as management staff members.

Our case base includes 150 cases of notebook computers described by eight attributes: *processor speed*, *memory capacity*, *hard disk capacity*, *display type*, *multimedia options* (CDROM, CDRW, etc.), *operating system*, *brand* and *price*. These cases include 19 processor speeds, 5 memory sizes, 6 hard disk sizes, 12 display types, 5 multimedia options, 4 operating systems, 7 brands, 8 price ranges (as shown in Table 6.1), and approximately 150 pictures of these notebook computers. There are four symbolic attributes: *display style*, *operating system*, *multimedia* and *brand*. The other attributes are numeric values. All these numeric attributes are discrete numbers, except for the price which is continuous.

6.1 Run-time Environment and Developing Tools

Application Server is a platform for designing, developing, debugging, distributing, implementing and managing an internet-based e-commerce application system. It is best suited for Business-to-Business (B2B) and Business-to-Consumer (B2C) e-commerce web sites. Certainly, any other applications of internet-based and

accessing data mainly by web browser also can apply Application Server techniques, such as electronic post office, bank transfer and search engine, etc. Compared to traditional techniques, Application Server techniques have extensibility and stability.

Table 6.1 Attributes and values of the notebook computers

Processor (MHz)	Memory (MB)	Harddisk (GB)	Display	Multimedia	OS	Brand	Price (CANS)
700	64	15	10.4"XGA TFT	None	Win 98	Accr	0-1000
750	128	20	11.3"XGA TFT	CDROM	Win 2000	Compaq	1001-1500
800	256	30	12.1"XGA TFT	CD-RW	Win XP Home	HP	1501-2000
850	512	40	13.3"XGA TFT	DVD	Win XP Pro	IBM	2001-2500
866	1024	48	14.1"XGA TFT	DVD/CD-RW		Panasonic	2501-3000
900		60	14.1" SXGA TFT			Sony	3001-3500
933			15"XGA TFT			Toshiba	3501-4000
1000			15" SXGA TFT				4001+
1130			15" UXGA TFT				
1200			15.1" XGA TFT				
1300			16.1" XGA TFT				
1330			16.1" UXGA TFT				
1400							
1500							
1600							
1700							
1800							
1900							
2000							

We believe that many open source products can provide reliable, scalable and secure solutions.

Apache is the most popular web server on the Internet since April of 1996. The December 2002 Netcraft Web Server Survey [Netcraft] found that 62.02% of the sites on the Internet use Apache, making it more widely used than all the other web servers combined.

Tomcat is the servlet container used in the official Reference Implementation for the Java Servlet and JavaServer Pages (JSP) [JSP] technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

JavaServer Pages technology allows web developers and designers to rapidly develop and easily maintain information-rich, dynamic web pages that leverage existing business systems. As part of the Java family, JSP technology enables the rapid development of web-based applications that are platform independent. JSP technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content.

JSP technology uses XML-like tags that encapsulate the logic that generates the content for the page. In addition, the application logic can reside in server-based resources (such as Java Beans component architecture) that the page accesses with these tags. All formatting (HTML or XML) tags are passed directly back to the response page. By separating the page logic from its design and display and by supporting a reusable component-based design, JSP technology makes it faster and easier than ever to build web-based applications.

A relational database is used to save our case base and other data. In order to emphasize our methodologies and algorithm, we selected Microsoft Access, which has a very friendly user interface like our database. However, any relational database can be used in our system, because *PCFinder* uses Structured Query Language (SQL) to access the database.

To implement our online computer store with *PCFinder*, we chose Apache Tomcat 4.0 as our Application Server; we use JavaServer Pages and Java Beans as our developing tools. We also apply XML as a Standard Generalized Markup Language, which is

transformed to HTML by the XSLT processor.

6.2 Main Class Diagrams of PCFinder

Our *PCFinder* includes four main modules, which are management, search, adaptation and clustering analysis. The following sections present the class diagram of each of the four modules.

6.2.1 Class Diagram of Management Module

Management module provides three functions: interacting with the user in the first tier, managing the other modules and maintaining the back-end databases. To perform these functions, there are 11 classes defined in this module. They are Configuration, CustomerQuery, Case, HistoricalPurchaseRecords, Management, DatabaseConnection, CustomerProfiles, CaseConstraints, ContactInfo, ProfileInfo and LongtermConstraints.

- Configuration: representing the computer configuration information.
- CustomerQuery: representing the customer query information getting from the first tier.
- Case: representing the cases information getting from the back-end database.
- HistoricalPurchaseRecords: representing the historical purchase records getting from the back-end database.
- Management: maintaining the back-end databases and managing the other modules.
- DatabaseConnection: setting up the connection with the back-end databases.
- CustomerProfiles: representing the customer profiles getting from the

back-end database.

- **CaseConstraints**: representing the case constraints getting from the back-end database.
- **ContactInfo**: representing the customers' contact information.
- **ProfileInfo**: representing the customers' profile information.
- **LongtermConstriants**: representing the customers' long-term constraints.

In these classes, Management class and DatabaseConnection class are control and boundary classes, which interface with other modules or databases. The other classes are entity classes, which are used to model information and associated behavior that must be stored. The following figure shows the relationship between these classes (see Figure 6.1).

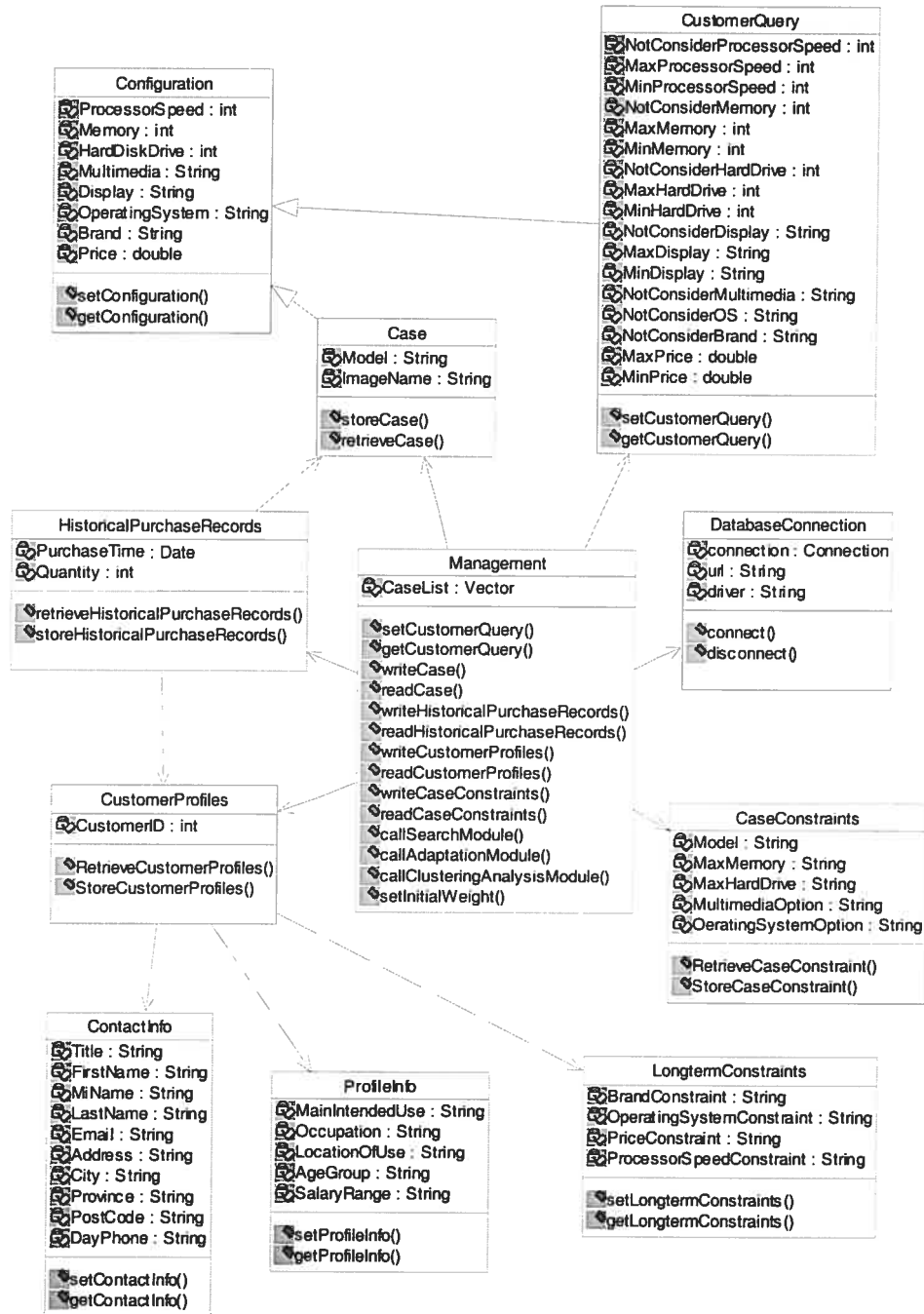


Figure 6.1 Class diagram of management module

6.2.2 Class Diagram of Search Module

Search module provides one function: finding out the most appropriate product from the back-end database according to the customer's query and long-term constraints. To perform this function, there are 6 classes used in this module. They are Configuration, CustomerQuery, Case, SelectionEngine, AttributeWeight and LongtermConstraints.

In these classes, SelectionEngine is a control and boundary class, which calculates the similarity between customer's query and case base, and finds the best-matched case. Class AttributeWeight is an entity class, which sets the weight of each attribute and modifies the weights according to the customer's query. The other classes have been described above. The following figure shows the relationship between these classes (see Figure 6.2).

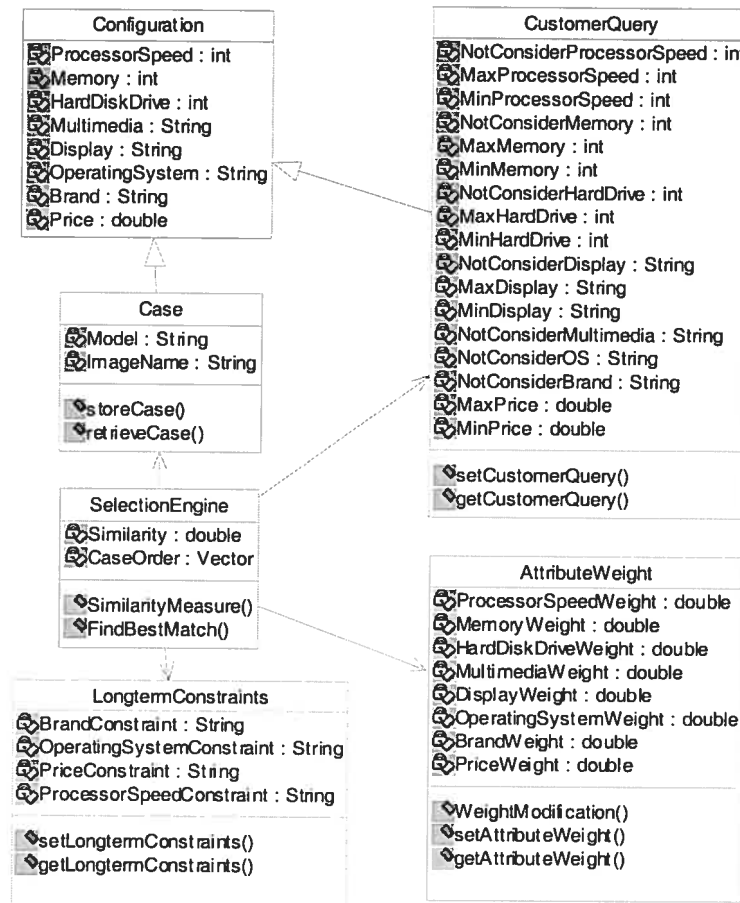


Figure 6.2 Class diagram of search module

6.2.3 Class Diagram of Adaptation Module

Adaptation module provides one function: modifying the product to better fulfill the customer's need according to the case constraints. To perform this function, 5 classes are used to describe this module. They are Configuration, CustomerQuery, Case, Adaptation and CaseConstraints.

In these classes, Adaptation is a control and boundary class, which checks the case constraints and adapts the case according to the customer's query. The other classes

have been described above. The following figure shows the relationship between these classes (see Figure 6.3).

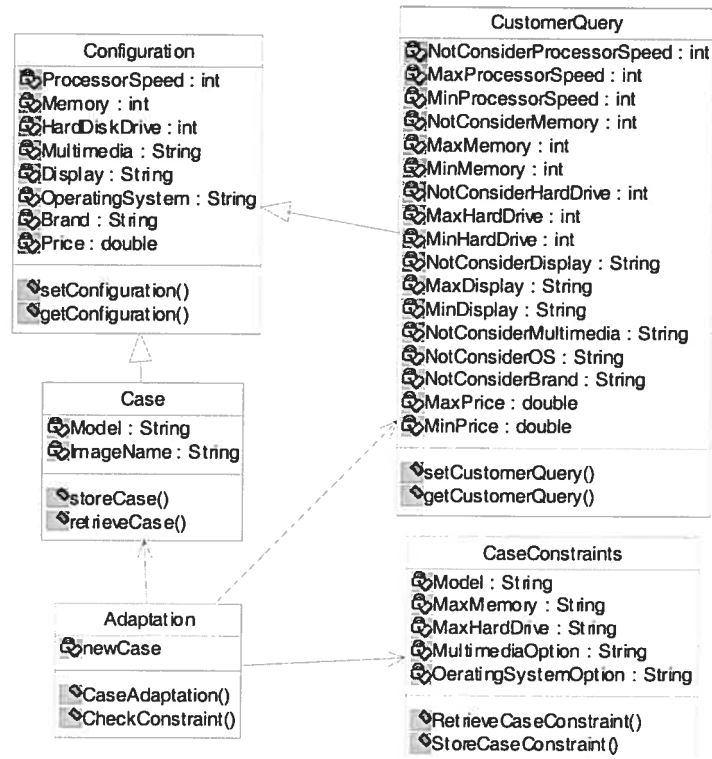


Figure 6.3 Class diagram of adaptation module

6.2.4 Class Diagram of Clustering Analysis

Clustering analysis module provides two functions: analyzing the information of customer profiles and historical purchase records and generating rules for management module to adjust some initial values. To perform this function, 9 classes are used to describe this module. They are Configuration, Case, ClusteringAnalysis, Rules, HistoricalPurchaseRecords, CustomerProfiles, ContactInfo, ProfileInfo and LongtermConstraints.

In these classes, ClusteringAnalysis is a control and boundary class, which groups the customer's profiles, analyzes historical purchase records according to the group of customer's profiles and generates the rules. Rules is an entity class, which sets the initial weight of each attribute. The other classes have been described above. The following figure shows the relationship between these classes (see Figure 6.4).

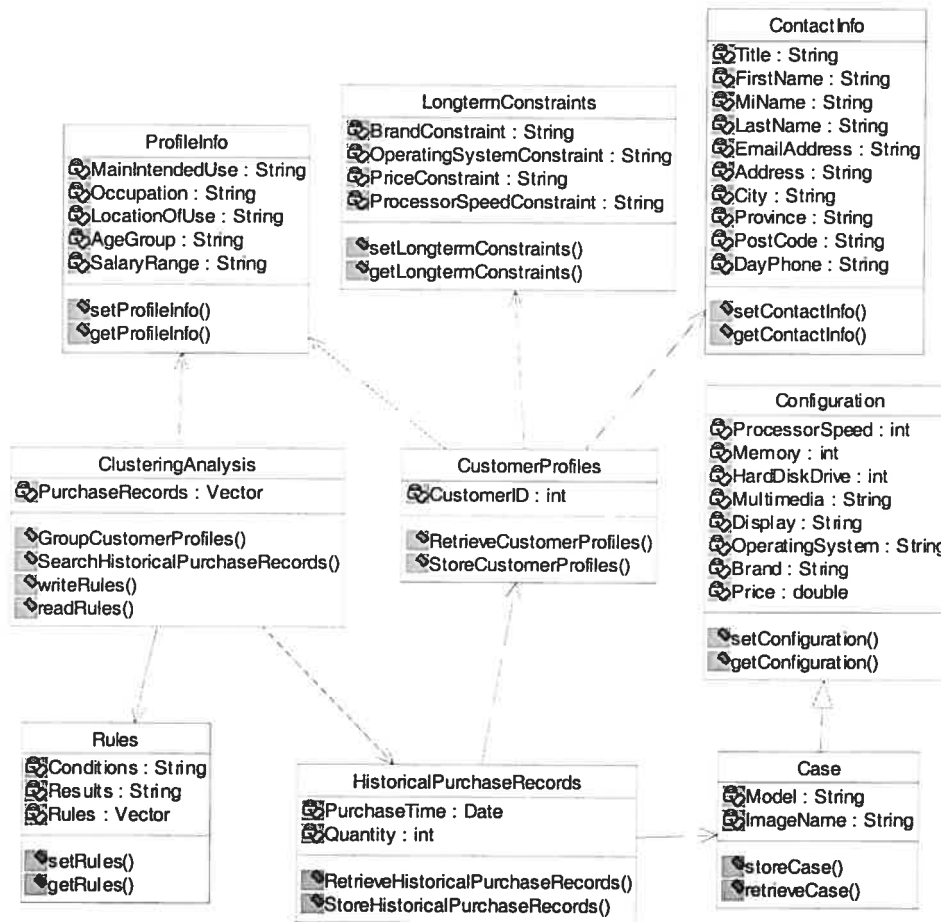


Figure 6.4 Class diagram of clustering analysis module

6.3 Main Scenario Diagrams of PCFinder

In the following sections, we use scenario diagram to show the shopping behavior of both unregistered users and registered users, and the administrator's management behavior.

6.3.1 Unregistered Users' Shopping Behavior Scenario

The following figure shows unregistered users' shopping behavior (see Figure 6.5). If step 3 is invoked, step 4, 5, 6, 7 and 8 will be skipped. Step 4 can be skipped according to the user's demand.

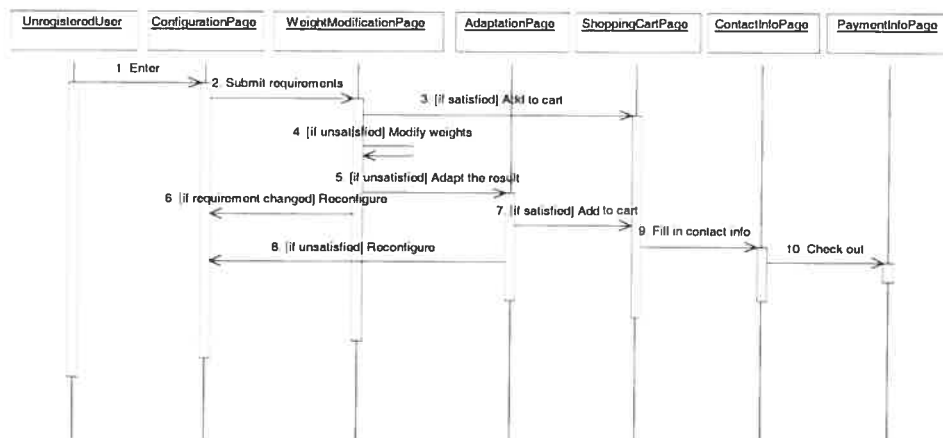


Figure 6.5 Scenario diagram of unregistered users' shopping behavior

6.3.2 Registered Users' Shopping Behavior Scenario

The following figure shows registered users' shopping behavior (see Figure 6.6). If step 5 is invoked, step 6, 7, 8, 9 and 10 will be skipped. Step 6 can be skipped according to the user's demand.

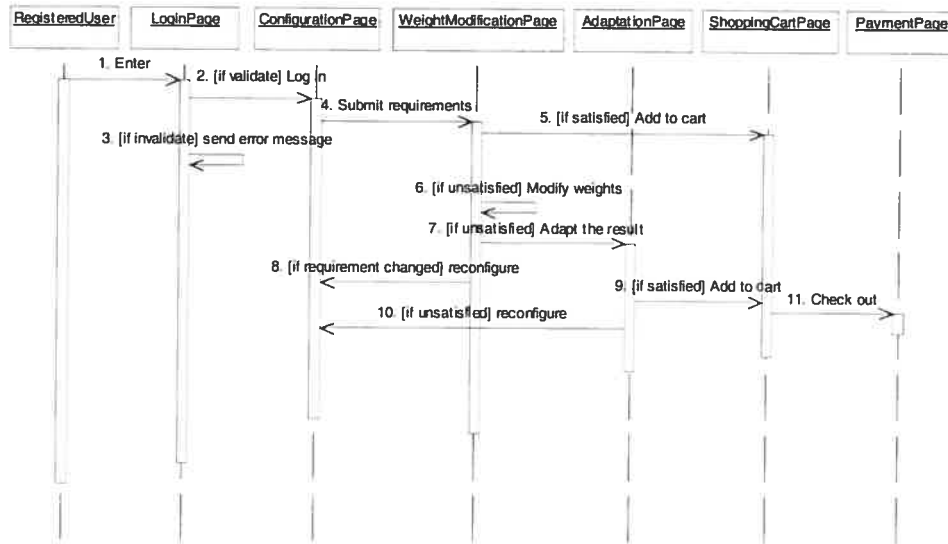


Figure 6.6 Scenario diagram of registered users' shopping behavior

6.3.3 The Administrator's Management Behavior Scenario

The following figure shows the administrator's management behavior (see Figure 6.7). Steps 4, 7 and 10 are optional. Either one or all of them can be invoked according to the administrator's demand.

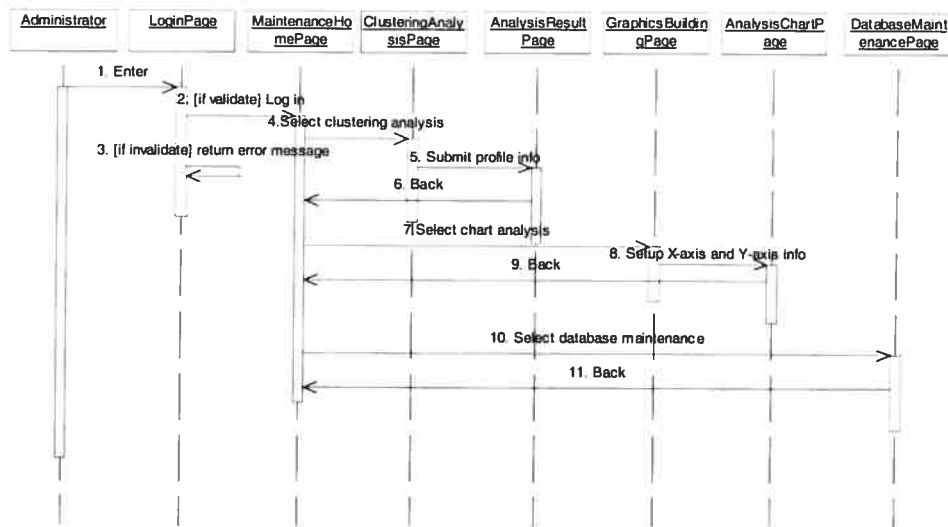


Figure 6.7 Scenario diagram of the administrator's management behavior

6.4 User Interface of PCFinder

In order to better represent *PCFinder*, the following sections will introduce the user interface of *PCFinder*, which includes main menu, customer shopping interface and administrator management interface.

6.4.1 Starting up PCFinder

To implement this product recommendation agent, we constructed an online computer store (See Figure 6.8). The main menu is separated into 4 sections. The first introduces *PCFinder* and its developers. The second part looks after the customer's account and includes "My Account Login", "Register Now", "My Cart" and "Order Status". The third part provides purchase assistance and includes "Notebook Finder", "Browse", "Payment Options", "Tax & Shipping Info", "Returns", "Online Security" and "Privacy Policy". The last part is reserved for administrators; it does not need to appear on this page, but we included it to make it simple to access.



Figure 6.8 Main page of the online notebook computer store

6.4.2 Customer Entry

Customers can access the web site as registered or unregistered users. If a customer registered, the system will assist the customer more effectively because it can remember the customer's personal information and historical purchasing records. Therefore customers who often shop on the site are recommended to register and login at each visit. Figure 6.9 shows the login page of this system.

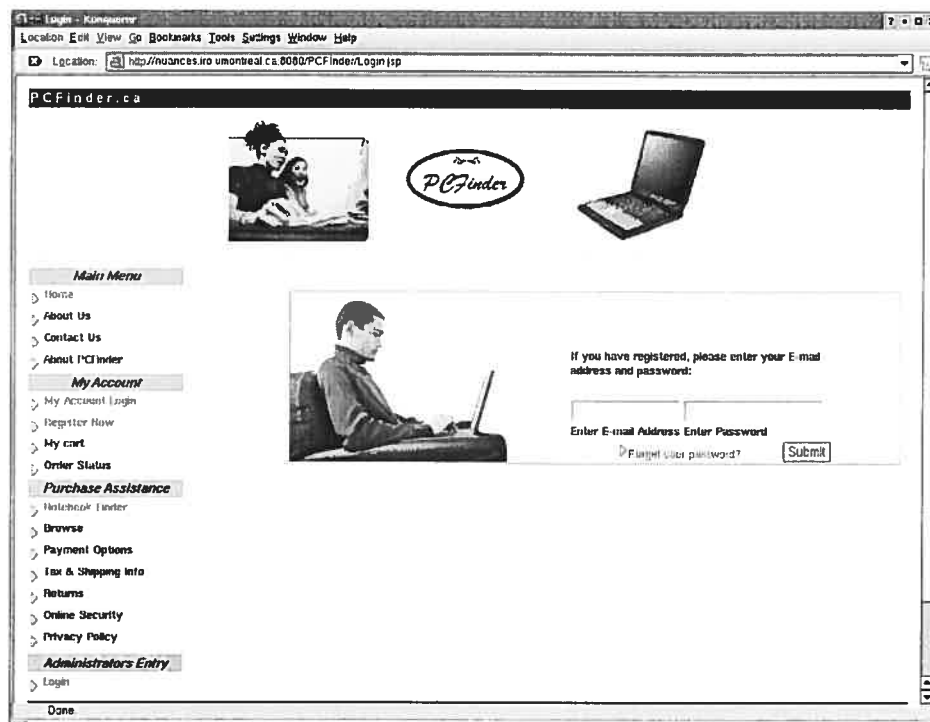


Figure 6.9 Login page of the online notebook computer store

6.4.2.1 Registration and Constraints Suggestion

When a new customer registers, the system asks him to fill out the form of required contact information (See Figure 6.10) and an optional profile information (See Figure 6.11). As a first response, the agent provides suggestions about the long-term constraints (See Figure 6.12) according to the profile information. If the customer is

not satisfied with these suggestions, he can modify the long-term constraints information by himself. We assume that customers who have the same profile will have the same preferences for computer features; therefore we apply collaborative-based filtering techniques to assist the customer. This makes the interaction more comfortable for the user, who does not need to fill in all these data.

Figure 6.10 Contact information form

The screenshot shows a web browser window with the URL `http://nuances.iro.umontreal.ca:8080/PCFinder/Register1.jsp`. The page features a navigation menu on the left and a 'Profile information' form on the right. The form contains several sections with radio button options:

- Main intended use of computer (Choose One):**
 - Software Development
 - Office Automation
 - Gaming
 - Corporate Applications
 - Internet Connectivity
 - Scientific Computing & Analysis
 - Graphics/Multimedia Design
- Occupation of user (Choose One):**
 - Student (K-12)
 - Student (University)
 - Government-Public sector
 - Education (K-12)
 - Education (University)
 - Self-employed
 - Corporate Employee
 - Unemployed
 - Others
- Location of use (Choose All Applicable):**
 - Home
 - Office
 - Travel - Airplane
 - Travel - Hotel
 - Travel - Train/Car
- Age Group of user:**
 - Under 20
 - 21-30
 - 31-40
 - 41-50
 - 51-60
 - Older 61
- Salary of user (if personal use.):**
 - 0-19,999
 - 20k-39,999
 - 40k-59,999
 - 60k-79,999
 - 80k+

A 'Continue' button is located at the bottom of the form.

Figure 6.11 Profile information form

The screenshot shows a web browser window with the URL `http://nuances.iro.umontreal.ca:8080/PCFinder/Register2.jsp`. The page features a navigation menu on the left and a 'Long-term constraints' form on the right. The form contains several sections:

- Long-term Constraints:**

According to PCFinder's analysis, some suggestions about the long-term constraints are shown below.
Note: These data are just suggestions, you can change them now.
- Which brand you never buy?**
 - Acer
 - Compaq
 - HP
 - IBM
 - Panasonic
 - SONY
 - Toshiba
- Which Operating System you are not interested in?**
 - Win 98
 - Win 2000
 - Win XP Home
 - Win XP Pro
- Tell us the price you usually prefer**

Minimum: CAN\$ Maximum: CAN\$
- Tell us the processor speed you usually prefer**

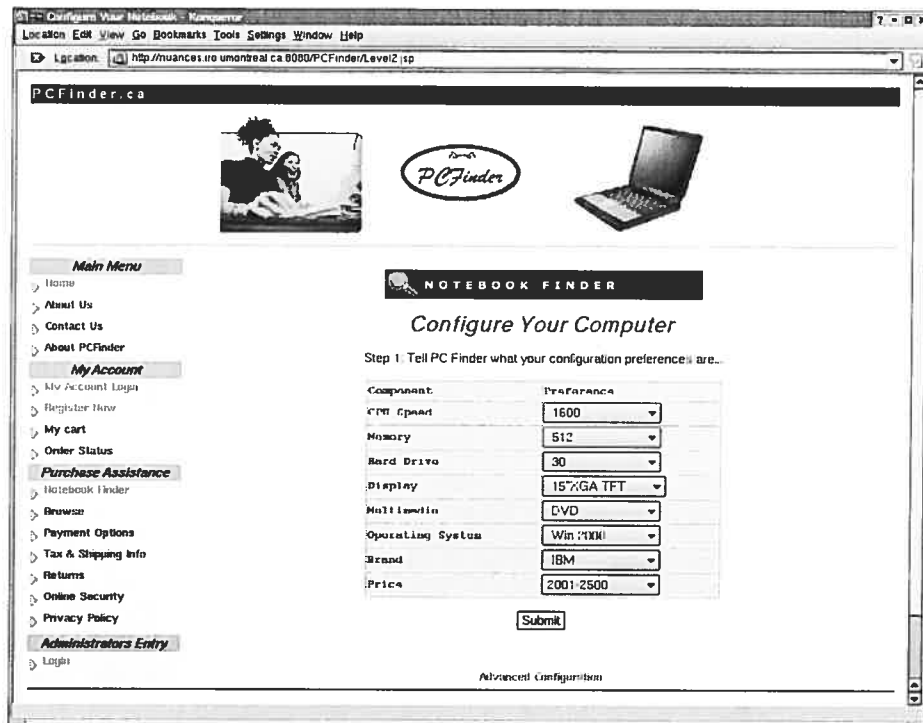
Minimum: MHz Maximum: MHz
- Do you want to save these changes?**
 - Yes
 - No

A 'Submit' button is located at the bottom of the form.

Figure 6.12 Long-term constraint form

6.4.2.2 Product Recommendation

In this section, we explain how the agent achieves the product recommendation process. We provide two kinds of user interfaces to allow the customer to configure her computer. The first is the basic configuration or basic search (See Figure 6.13). The other is the advanced configuration or advanced search (See Figure 6.14). In the first case, the user only specifies to the agent the preferred value of the attribute; in the second, the user specifies the preferred value, a value to exclude, and a range of values that can be considered.



The screenshot shows a web browser window displaying the PCFinder website. The browser's address bar shows the URL <http://muance1.no.umontreal.ca/8080/PCFinder/Level2.jsp>. The website header includes the PCFinder logo and a navigation menu. The main content area is titled "NOTEBOOK FINDER" and "Configure Your Computer". Below this, there is a form for configuring a notebook computer. The form is titled "Step 1: Tell PC Finder what your configuration preferences are..." and contains a table with the following components and preferences:

Component	Preference
CPU Speed	1600
Memory	512
Hard Drive	30
Display	15" XGA TFT
Multimedia	DVD
Operating System	Win 7/8/10
Brand	IBM
Price	2001-2500

Below the table is a "Submit" button. At the bottom of the form, it says "Advanced Configuration".

Figure 6.13 Basic configuration form

Component	Preference	Not Consider	MAXIMUM	MINIMUM
CPU Speed	1600	(Select from list)	1800	1200
Memory	512	(Select from list)	(Select from list)	256
Hard Drive	30	(Select from list)	40	20
Display	15"XGA TFT	(Select from list)	16.1"XGA TFT	14.1"XGA TFT
Multimedia	DVD	CDROM		
Operating System	Win 2000	Win 98		
Brand	IBM	Panasonic		
Price	2001-2500		(Select from list)	(Select from list)

Figure 6.14 Advanced configuration form

When the desired computer has been configured, the agent finds the five most similar computers in the product cases according to Case-Based Reasoning theory and recommends them to the customer. The result is shown in Figure 6.15. Then the agent interacts with the customer. For research reasons, the total similarity, the local similarity and the weight of the attribute can be displayed or hidden in each step of the interaction between the agent and the customer (See Figure 6.18).

The agent asks the customer if he is satisfied with the recommended computer (See Figure 6.16). If yes, the customer can add this computer to his shopping cart (See Figure 6.17), and the recommendation phase is finished. Otherwise, the customer can refine the result.

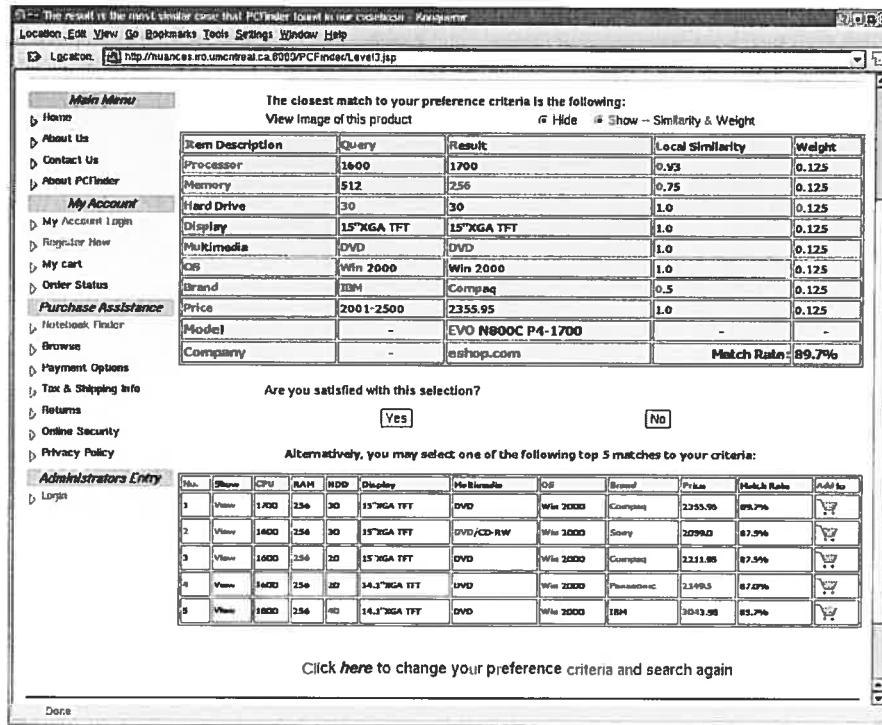


Figure 6.15 The five most similar computers

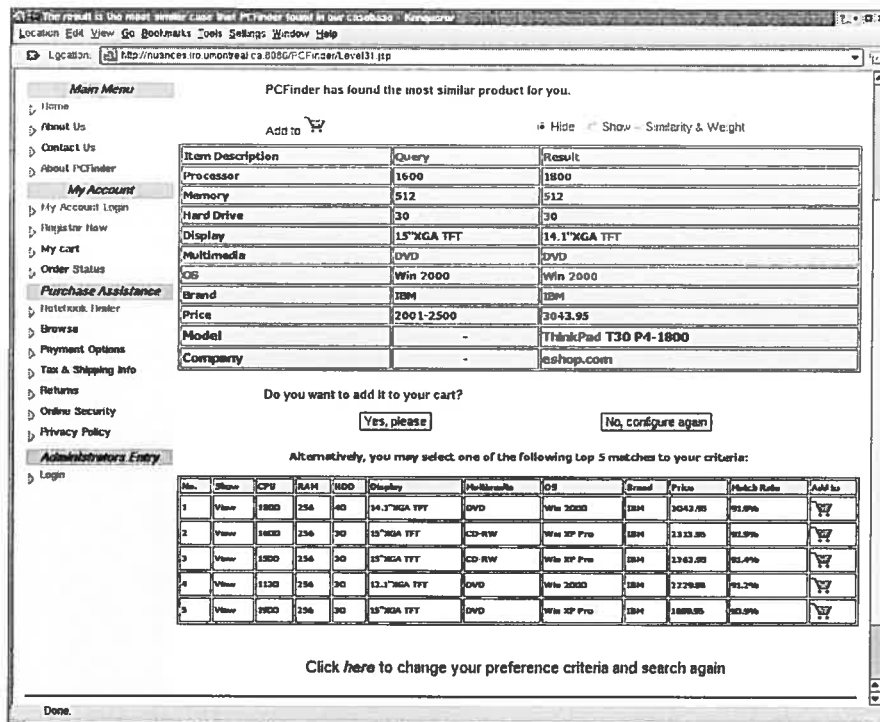


Figure 6.16 Interaction with the customer: add to cart or not?

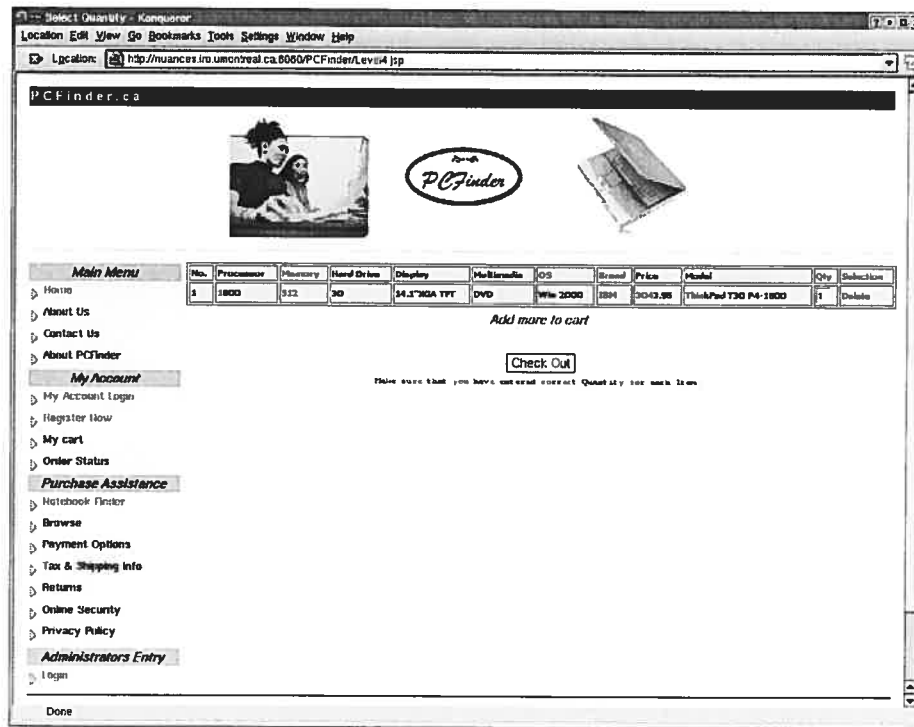


Figure 6.17 Shopping cart

PCFinder provides two ways to assist the customer in refining the result. The first way is weight modification. The product consists of several attributes. Different attributes have different weight in the customer's mind. According to CBR theory, the product is recommended with respect to the integration of the similarities of the whole attributes. One of the possibilities occurs if the most important attribute has the same weight as the other attributes at a time when the customer is unsatisfied with the result. In this case, weight modification can be applied to help the customer find a more satisfactory solution. Consider the situation illustrated in Figure 6.18, for instance. The solution proposed as "result" is the closest in similarity to the customer's "query" among the 150 cases in the case base. But the customer declares himself unhappy with the switch from IBM to Compaq. Therefore, *PCFinder* reduces the weights of all the attributes that had a local similarity higher than that of the brand (in this case, all the attributes) and increases the weight of "brand". The case base is searched again with these new weights and the best match that is found is illustrated in Figure 6.19.

Another way to increase the customer's satisfaction is adaptation of the result. Some attributes of a product can be adapted, others cannot. Therefore, if the customer is still not satisfied with the attributes, some of them can be adapted. In this case, *PCFinder* helps the customer in adapting the recommended computer until he is satisfied. In our system, the attributes of memory, hard drive, multimedia and operating system can be adapted. Figure 6.20 shows the result of adaptation starting from the unsatisfactory solution that was previously offered in Figure 6.19. It is important to point out that this is the only time that the system allows itself to recommend a product that may not be in the case base, which explains why a solution so close to the customer's query had not been proposed earlier.

If, after these two phases, the customer has found a satisfactory computer, he can add it to his shopping cart and the recommendation phase is finished.

Before selecting the "Check Out" button, unregistered customers must fill out a contact information form. The agent fills it out automatically for registered customers. After the customer selects a payment option, the agent reminds the customer when he will receive the products and thanks for his purchase (See Figure 6.21).

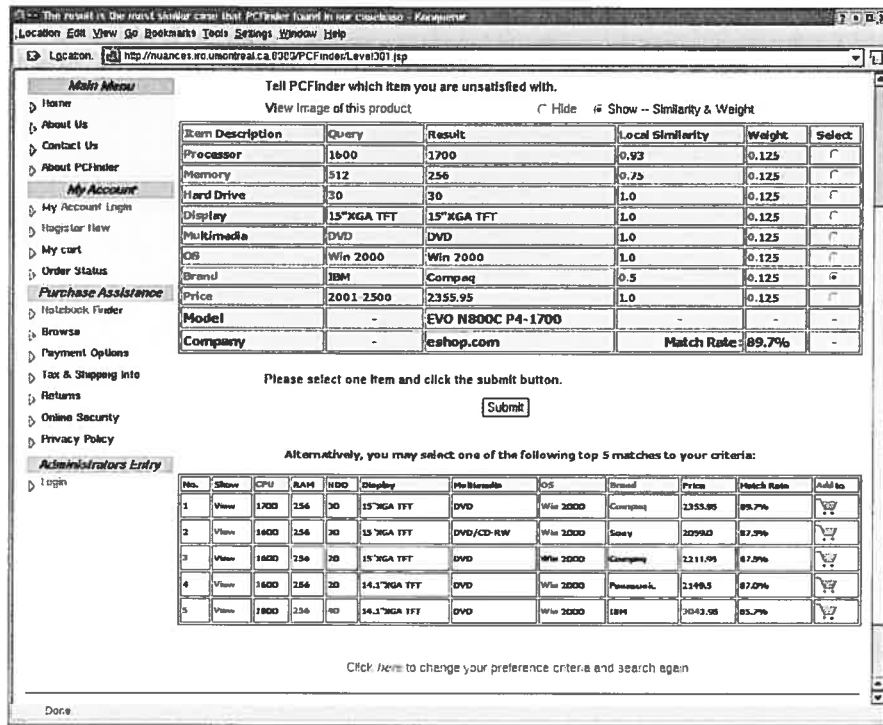


Figure 6.18 Before weight modification

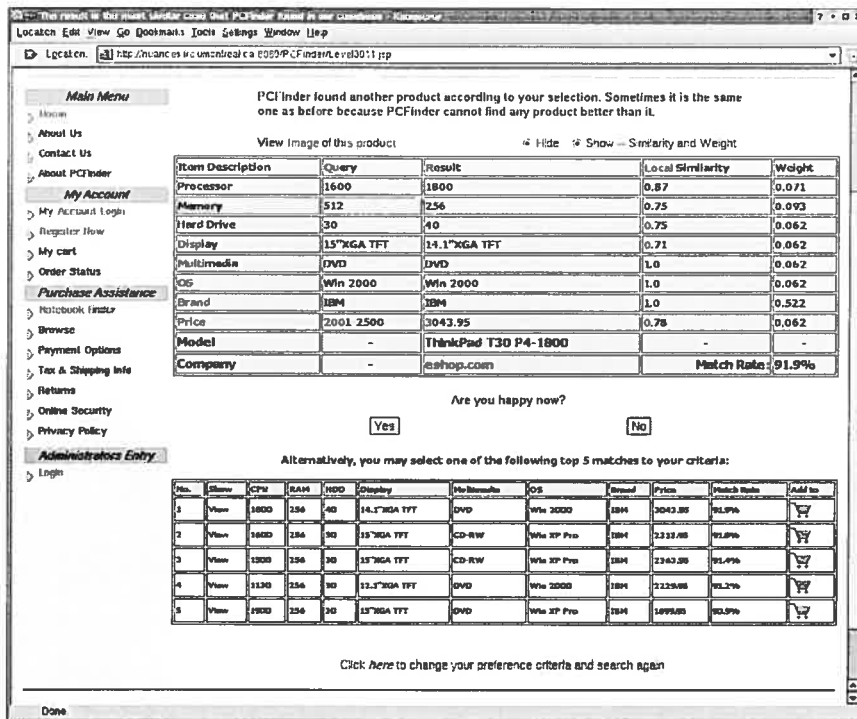


Figure 6.19 After weight modification

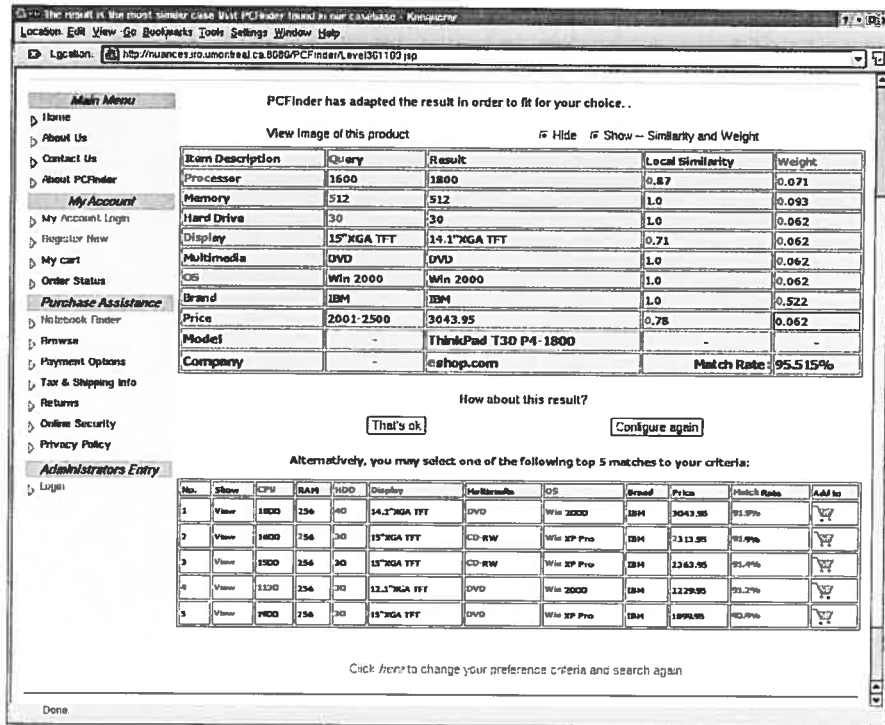


Figure 6.20 After adaptation of the result

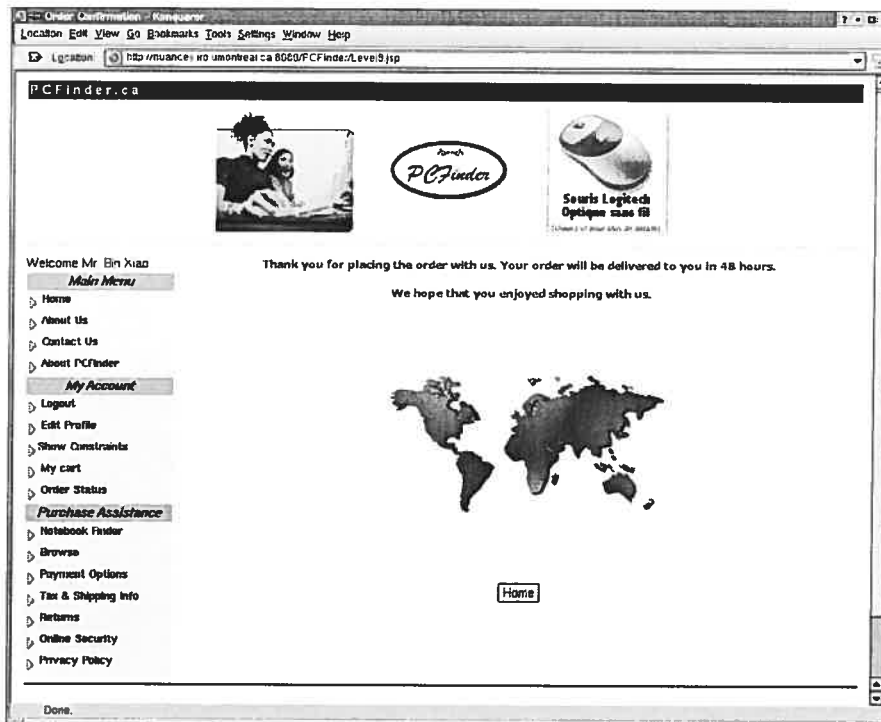
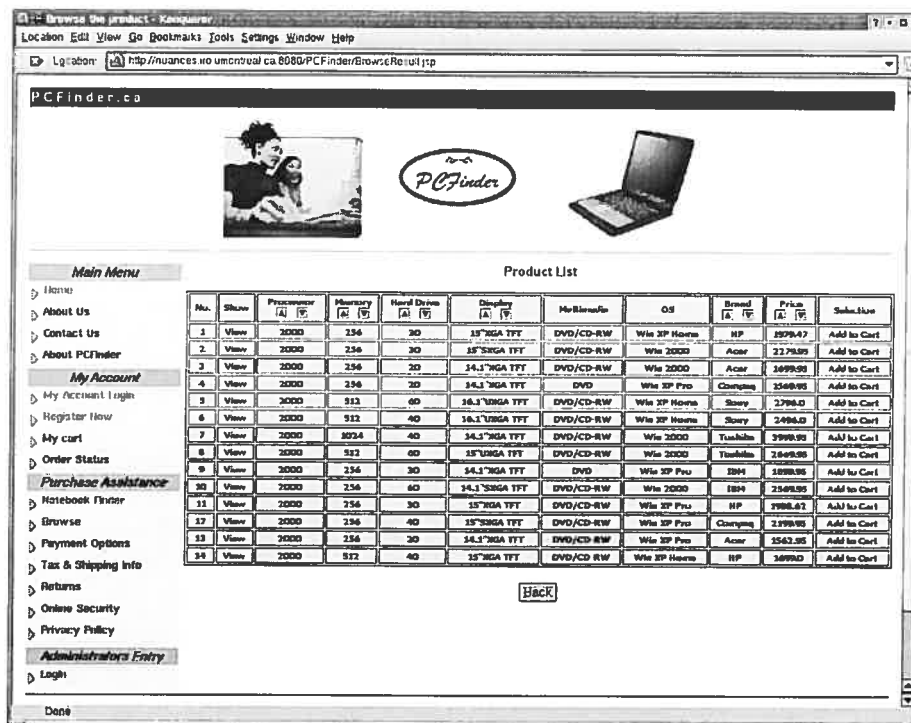


Figure 6.21 Buying session finished

6.4.2.3 Browsing Function and Password Reminder

Our online computer store also provides the customer with a browsing function to find a pre-configured computer. The customer may browse through all products or browse through products according to processor speed or brand. The product list returned can also be ordered according to processor speed, memory size, hard drive capacity, display type, brand or price (See Figure 6.22).

The system also offers a password reminder service, which prompts the user for the answers to a set of questions entered during registration (See Figure 6.23).



The screenshot shows a web browser window displaying the PCFinder website. The page features a navigation menu on the left and a main product list table. The table contains 14 rows of computer configurations, each with a 'View' link and an 'Add to Cart' link. The columns in the table are: No., Show, Processor, Memory, Hard Drive, Display, HardWare, OS, Brand, Price, and Sub-link.

No.	Show	Processor	Memory	Hard Drive	Display	HardWare	OS	Brand	Price	Sub-link
1	View	2000	256	30	15" WGA TFT	DVD/CD-RW	Win XP Home	HP	899.47	Add to Cart
2	View	2000	256	30	15" WGA TFT	DVD/CD-RW	Win 2000	Acer	2279.89	Add to Cart
3	View	2000	256	30	14.1" WGA TFT	DVD/CD-RW	Win 2000	Acer	1699.99	Add to Cart
4	View	2000	256	30	14.1" WGA TFT	DVD	Win XP Pro	Compaq	1569.99	Add to Cart
5	View	2000	512	60	16.1" WGA TFT	DVD/CD-RW	Win XP Home	Sony	2799.0	Add to Cart
6	View	2000	512	40	16.1" WGA TFT	DVD/CD-RW	Win XP Home	Sony	2499.0	Add to Cart
7	View	2000	8024	40	14.1" WGA TFT	DVD/CD-RW	Win 2000	Toshiba	2999.99	Add to Cart
8	View	2000	512	60	15" WGA TFT	DVD/CD-RW	Win 2000	Toshiba	2649.99	Add to Cart
9	View	2000	256	30	14.1" WGA TFT	DVD	Win XP Pro	Sony	1899.99	Add to Cart
10	View	2000	256	60	14.1" WGA TFT	DVD/CD-RW	Win 2000	IBM	2599.99	Add to Cart
11	View	2000	256	30	15" WGA TFT	DVD/CD-RW	Win XP Pro	HP	2998.61	Add to Cart
12	View	2000	256	40	15" WGA TFT	DVD/CD-RW	Win XP Pro	Compaq	2999.99	Add to Cart
13	View	2000	256	30	14.1" WGA TFT	DVD/CD-RW	Win XP Pro	Acer	2542.95	Add to Cart
14	View	2000	512	40	15" WGA TFT	DVD/CD-RW	Win XP Home	HP	1899.0	Add to Cart

Figure 6.22 Browsing by processor speed

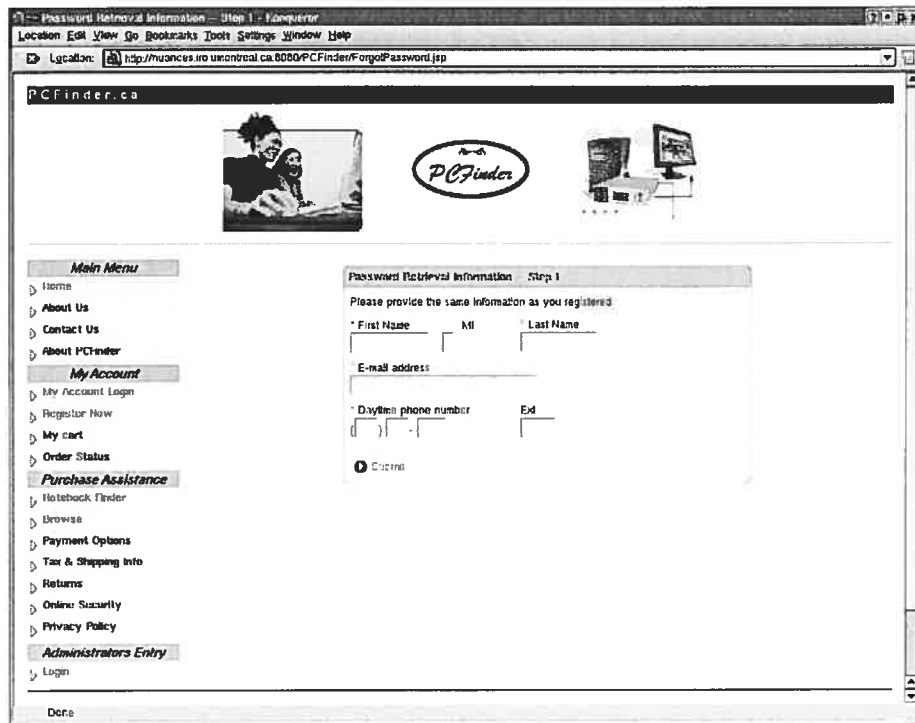


Figure 6.23 Password retrieval – step one

6.4.3 Administrator Entry

Our system provides some analysis and maintenance tools for administrators and management staff (See Figure 6.24). The analysis tools include a cluster analysis tool and a graphic-building wizard. The maintenance tools are tools to review records of past purchase and to add, modify or delete a significant case.

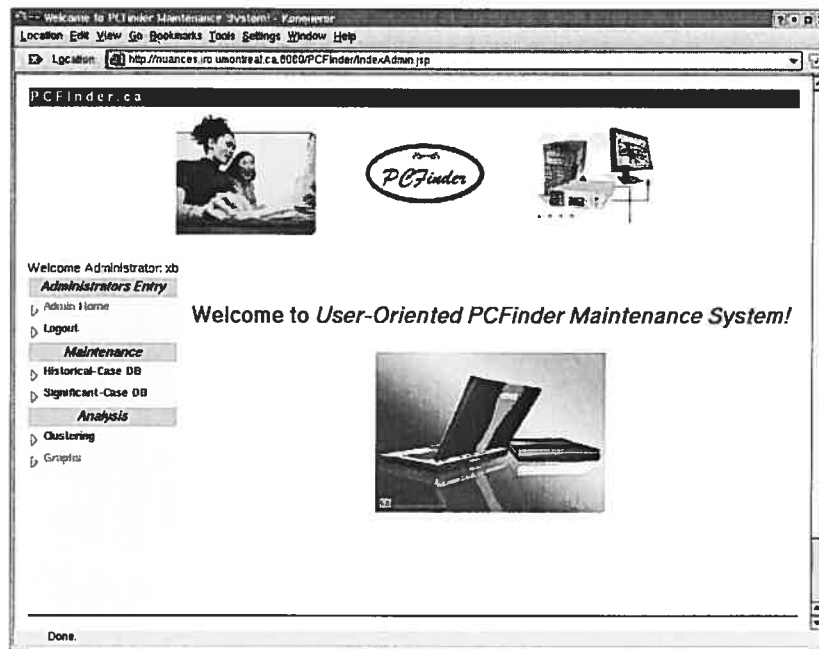


Figure 6.24 Maintenance main page

6.4.3.1 Cluster Analysis

The cluster analysis tool analyzes the user profiles and provides suggestions for the configuration of the notebook computer. A customer's profile includes the intended use for the computer, the location of use (e.g. home, office, etc.), and personal information about the user such as age, occupation, and salary range (See Figure 6.25). All of these questions take single-selection answers except for the location of use. Based on the user's form, the cluster analysis tool recommends a computer configuration based on the configurations of a group of previous consumers with the same background. It returns the most frequently selected configurations of this group, including brand, operating system, processor speed, memory capacity, hard drive capacity, multimedia option (e.g. DVD), and display system (e.g. size and type.). See Figure 6.26 (For the processor speed, it provides both the most frequently selected processor speed and the highest speed selected.). The tool also returns the range of prices paid by this group.

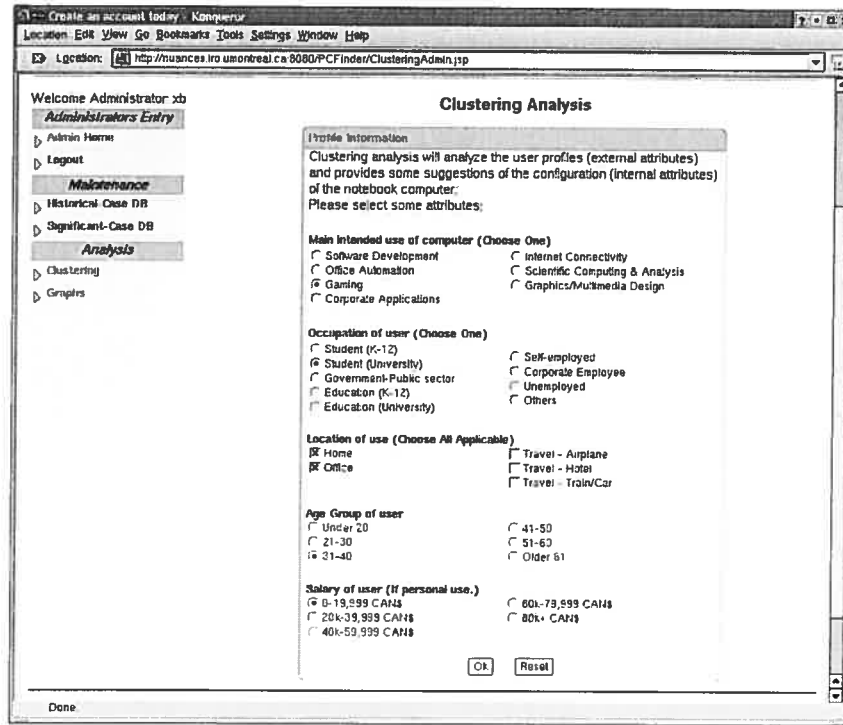


Figure 6.25 User profile form

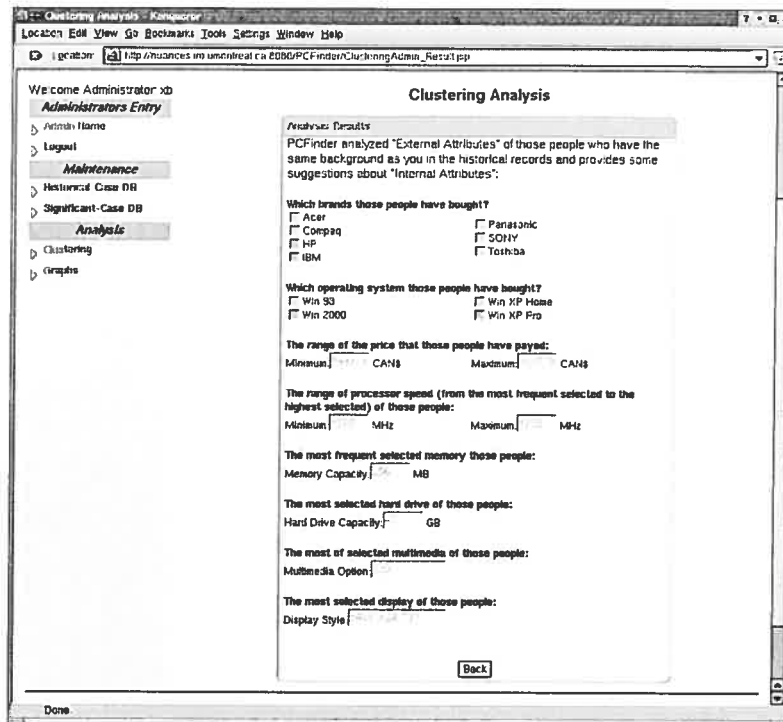


Figure 6.26 Configuration after clustering analysis

6.4.3.2 Graphical Analysis

To visualize the relationship between user profiles (external attributes) and the product attributes (internal attributes), we construct a graphic-building wizard for management staffs' marketing research. After selecting any items (one or more) of the user profiles and any one attribute of the product (See Figure 6.27), we get a statistical diagram (See Figure 6.28).

The X-axis represents the selected attribute of the product. The Y-axis represents the quantity of sales. There are two curves in this diagram. For example, the selected item is playing games (the intended use of the computer is to play games); the selected attribute of the product is processor speed. This chart compares the sales between computers intended for playing games and not. According to this analysis, it is easy to know which attribute is more important for the customers in a specific group. Hence, the initial weight value or the recommended product attribute will be modified.

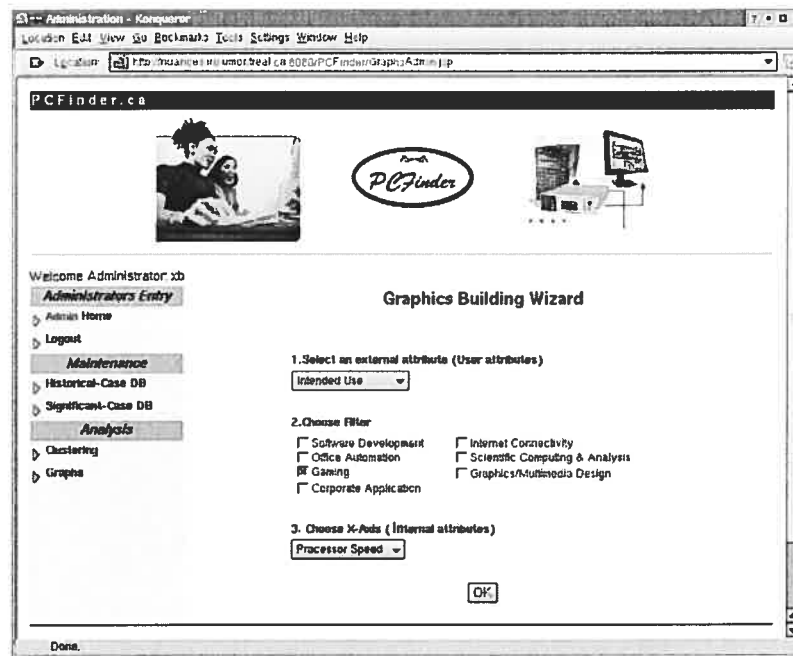


Figure 6.27 User interface of graphic-building wizard

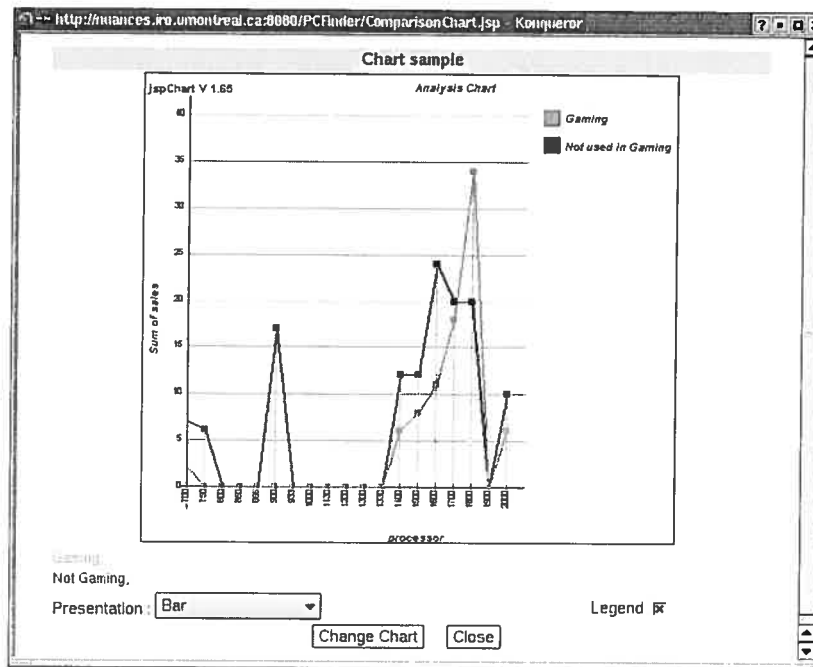


Figure 6.28 Statistical diagram generated by the graphic-building wizard

6.4.3.3 Maintenance

The main function of the purchase record maintenance tool is to review and check what happened in the purchase history. This database is strictly for view; it cannot be modified for anybody. The user can browse through all the products or can select products based on attribute values like brand or processor speed (See Figure 6.29). The browsing result includes customer information, significant case information, new case information, the quantity of sales, and the purchase date (See Figure 6.30).

Significant case maintenance options include adding, modifying and deleting a case (See Figure 6.31). There are two ways to find a case to be modified or deleted. The first is to enter the case ID number; the other is to select a case from the case list, which is generated by searching a keyword.

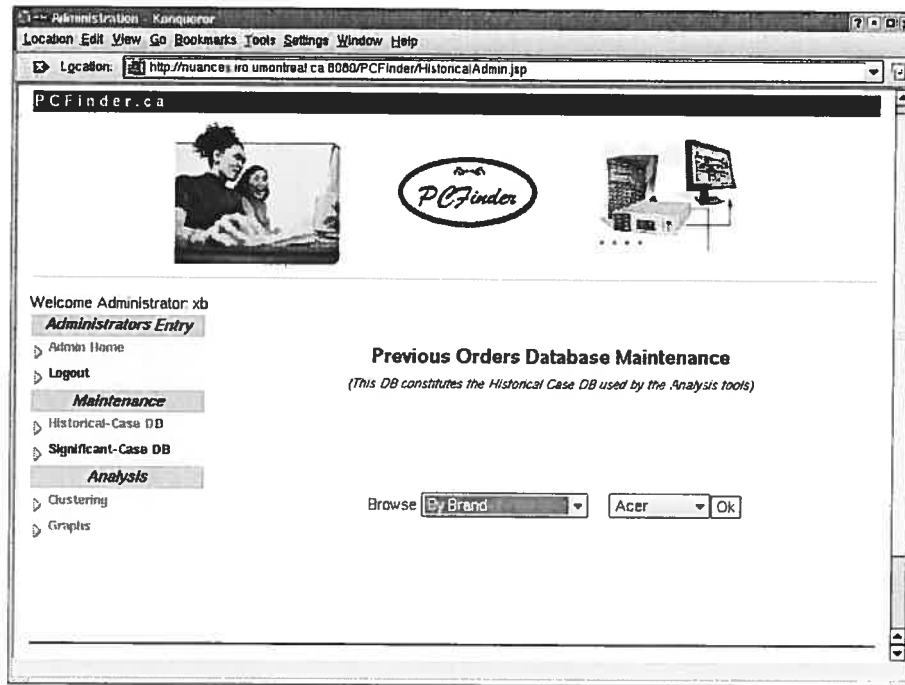


Figure 6.29 Browsing selection of the purchase records

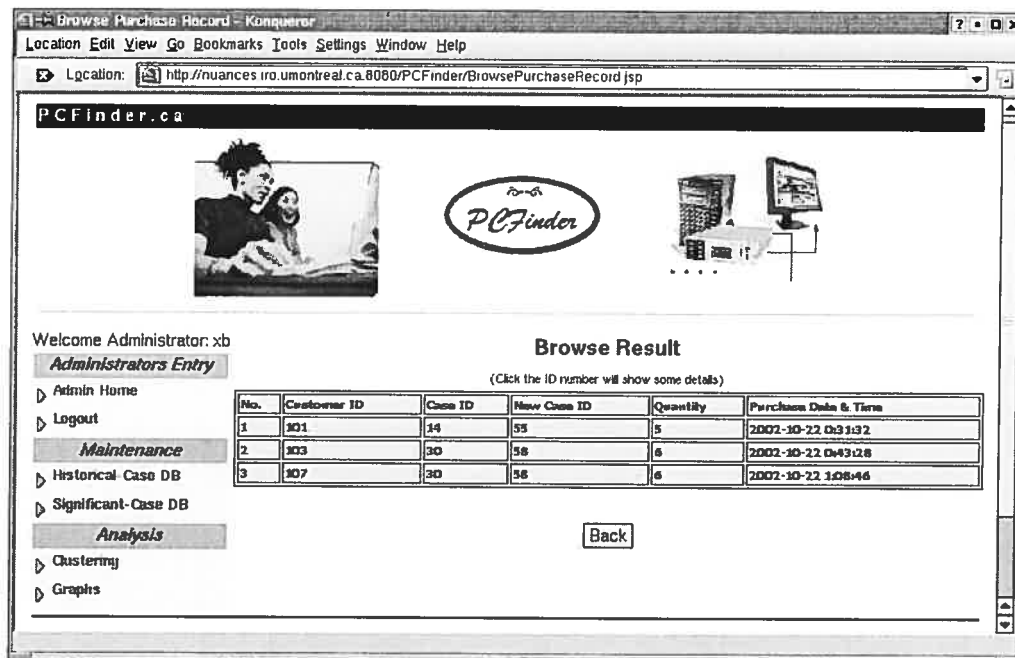


Figure 6.30 Browsing result of the purchase records

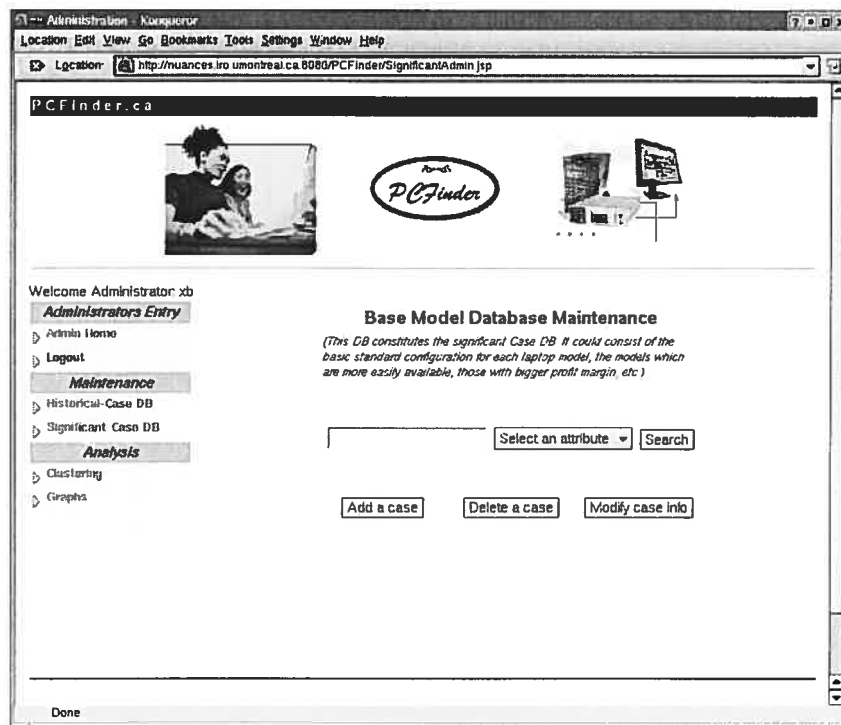


Figure 6.31 User interface of significant case maintenance tool

In this chapter, we represented the implementation of *PCFinder*, which includes the run-time environment and developing tools, class diagrams of main modules, scenario diagrams of users' and the administrator's behavior, and the user interface of *PCFinder*. In the following chapter, we will introduce the experiment and evaluation of our work.

Experiment and Evaluation

Since product recommendation systems are designed to improve the customer satisfaction, our measurement methods consist of asking customers about their satisfaction. In order to test the validity of the methods used in our *PCFinder*, an experiment was carried out. In this experiment, 36 people (of professional and non-professional backgrounds) were invited to use the system and submit their feedback. By analyzing this feedback, we were able to draw conclusions about the performance of our methods.

This experiment had five goals:

- To measure user satisfaction on the Order-Based Similarity Measure;
- To measure user satisfaction on the weight modification method;
- To measure user satisfaction on the adaptation method;
- To measure user satisfaction on the short-term profile using the advanced configuration;
- To measure user satisfaction on the long-term profile by comparing the recommended product obtained using the search with constraints with that obtained using the search without constraints.

7.1 Experimental Process and Methods

Our experimental method is based on user feedback. Therefore, 36 users with

professional and non-professional backgrounds were invited to run the system and then fill out a test form.

In order to test the performance of the approaches mentioned above, we constructed the following three test cases:

Case 1: Unregistered users using the basic configuration (See Table 7.1).

Case 2: Unregistered users using the advanced configuration (See Table 7.2).

Case 3: Registered users searching with constraints using the basic configuration (See Table 7.2).

Table 7.1 Test form for Case 1

Step 1. After simple searching	Order by agent	1	2	3	4	5
	Order by user	2	1	3	4	5
	Satisfaction rate	6				
Step 2. After weight modification	Order by agent	1	2	3	4	5
	Order by user					
	Satisfaction rate					
Step 3. After adaptation	Satisfaction rate					

Table 7.2 Test form for Case 2 and Case 3

Order by agent	1	2	3	4	5
Order by user					
Satisfaction rate					

In the evaluation form, “Order by agent” means the ranking of the top five matches according to the agent. “Order by user” means the ranking of the top five matches according to the user. “Satisfaction rate” rates the user’s satisfaction with the recommended result; it is a linear seven-point scale from 1 to 7. An extremely satisfied user will enter a value of 7, while a dissatisfied user will enter a value of 1.

For sake of the illustration, we gave in Table 7.1 an example of how the user could fill in the requested information after Step 1.

7.2 Evaluation Methods

Two evaluation methods were used in our experiment. The first uses a modified “*Euclidean Distance*” to evaluate the similarity between the recommended results provided by *PCFinder* and the evaluated results provided by the users. It serves to measure the degree of user satisfaction with the recommended results. The other method uses inference from large samples to evaluate the degree of user satisfaction. It indicates the performance of our methods such as the similarity measure, weight modification, adaptation, short-term profiles and long-term profiles.

Method 1: Evaluation using Euclidean distance

Euclidean distance is widely used to measure the similarity between two vectors in the area of Information Retrieval [Schroeder & Noy, 2001]. The formula for Euclidean distance is given by:

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

In this formula, A and B represent two vectors. In our case, we are comparing “Order by agent” and “Order by user”, which are five dimensional vectors. Vector A represents the evaluated order according to the user, such as (2, 1, 3, 4, 5). Vector B represents the order provided by the agent, namely (1, 2, 3, 4, 5).

However, since the vector elements represent a ranked order, the position of the

elements is very important. For example, the vectors (2, 1, 3, 4, 5) and (1, 2, 3, 5, 4) are at the same Euclidean distance from the vector (1, 2, 3, 4, 5). However, the user feels that these two evaluated orders are different. Hence, a new formula using the normalized Euclidean distance and taking into account the order of the vector elements is needed [Ma, 2001]:

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

The similarity falls in the range $0 \leq Sim_{AB} \leq 1$. In this formula, Sim_{AB} represents the similarity between the two vectors A and B . In our case, B represents the vector “Order by agent”, which is always (1, 2, 3, 4, 5), and A represents the vector “Order by user”, which is the sequence reordered by the user. The weight w_i of the i^{th} vector has a value between 1 and 5. The product recommended first, which is the first element in the vector, is given a weight of 5. The product recommended second, which is the second element in the vector, is given a weight of 4, and so on. The number $n=5$ of elements in the vector is the total number of recommended products.

Based on the normalized Euclidean distance Sim_{AB} , we define a *three-point scale* to evaluate the similarity. A similarity range of 0~0.6 is interpreted to mean the user is not satisfied with the recommended results, a range of 0.6~0.8 corresponds to an average satisfaction, and a range of 0.8~1.0 indicates an excellent match for the user’s request. These cut-off points are justified by looking at all 120 possible user orderings of 5 products. We find that only 11 orderings enjoy a similarity of 0.8 or higher, whereas roughly half the remaining orderings (56 to be exact) obtain a similarity between 0.6 (included) and 0.8 (excluded). The remaining 53 orderings fail with a

similarity smaller than 0.6. Notice however that the similarity, as computed with our formula, is never very close to zero: the smallest similarity occurs with user ordering $B=(5, 4, 3, 1, 2)$, in which case $Sim_{AB} = 1 - (41/125)^{1/2} \approx 0.427$.

Using this method, we compare the similarity between the order given by the agent and the order given by the users so that we can evaluate the performance of the Order-Based Similarity Measure.

Method 2: Evaluation using large-sample test of hypotheses

In this approach, we apply the method of large-sample statistical test to evaluate user satisfaction rates. We have applied two test models.

The first one is a one-tailed large-sample statistical test for a population mean [Mendenhall, 1983]. Such tests are considered to be appropriate provided the n observations in the sample were randomly selected from the population and n is large enough, say $n \geq 30$. Thus, we proceed with confidence since we have $n=36$. A summary of the test is shown below

1. *Null Hypothesis*: $H_0: \mu = \mu_0$
2. *Alternative Hypothesis (One-Tailed Test)*: $H_a: \mu > \mu_0$
3. *Test Statistic*:
$$z = \frac{\bar{y} - \mu_0}{\sigma_{\bar{y}}} = \frac{\bar{y} - \mu_0}{\sigma/\sqrt{n}}$$

Where \bar{y} is the sample mean; n is sample size; and σ is the standard deviation, which can be approximated by the sample standard deviation when σ is unknown.

4. *Rejection Region*: $z > z_\alpha$, where α is the tolerated error probability.

Using this statistical test, we analyze whether or not most testers are satisfied with our method of Order-Based Similarity Measure. For this purpose, we compute the probability α that the null hypothesis be valid according to the available data. This probability will be labeled *P-value* and estimated from the computed *z-value* using standard statistical tables of normal curve areas. We hope to find α as small as possible because rejection of the null hypothesis gives statistical evidence for the alternative hypothesis, which means that the average user satisfaction μ is indeed better than μ_0 . For the purpose of this study, we use $\mu_0 = 4$, which is the middle point in our seven-point scale for user satisfaction.

The second model is a one-tailed large-sample statistical test of a hypothesis about the difference between two population means, which is based on independent random samples [Mendenhall, 1983]. In this case, μ_1 denotes the average user satisfaction at some point in the recommendation process and μ_2 denotes the average user satisfaction at some later point. We wish to give statistical evidence that $\mu_2 > \mu_1$, which means that the user satisfaction has increased. For this purpose, we wish to reject the null hypothesis that $\mu_1 = \mu_2$ with a one-tailed test. (The standard procedure is to test if the difference between μ_1 and μ_2 is equal to some value D_0 , which is here set to zero, hence we write $\mu_1 = \mu_2$ instead of $\mu_1 - \mu_2 = 0$, etc.). Again, this approach is appropriate because $n=36$ is large enough. A summary of the test is shown below.

1. *Null hypothesis*: $H_0: \mu_1 = \mu_2$
2. *Alternative Hypothesis (One-Tailed Test)*: $H_a: \mu_2 > \mu_1$

$$3. \text{ Test Statistic: } z = \frac{\bar{y}_1 - \bar{y}_2}{\sigma_{(\bar{y}_1 - \bar{y}_2)}} = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

Where \bar{y}_1 and \bar{y}_2 are the sample means; n_1 and n_2 are the sample sizes; and σ_1^2 and σ_2^2 are the variances, which can be approximated by the sample variances when σ_1^2 and σ_2^2 are unknown.

4. *Rejection Region:* $z > z_\alpha$, where α is the tolerated error probability.

According to this statistical test, we analyze the differences of user satisfaction degrees between Order-Based Similarity Measure and weight modification, weight modification and adaptation, users with short-term profiles and users without them, and users with long-term profiles and users without them.

7.3 Evaluation Results

In this section, we present the results of our comparison of the agent-recommended and user-evaluated product rankings for each of the three test cases.

Using the first method, we computed the similarities between the agent and user orders, and then we compared the percentage of similarities according to the three-point scale (See Figure 7.1). We find that 38.9% of similarities are in the range 0.8~1.0, which is considered to be an excellent match, 50.0% of similarities are in the range 0.6~0.8, which is considered to be an average match, and only 11.1% of similarities are in the range 0.0~0.6, which is considered to be a poor match. Therefore, 88.9% of similarities are satisfactory or better. This illustrates the performance of the Order-Based Similarity Measure algorithm.

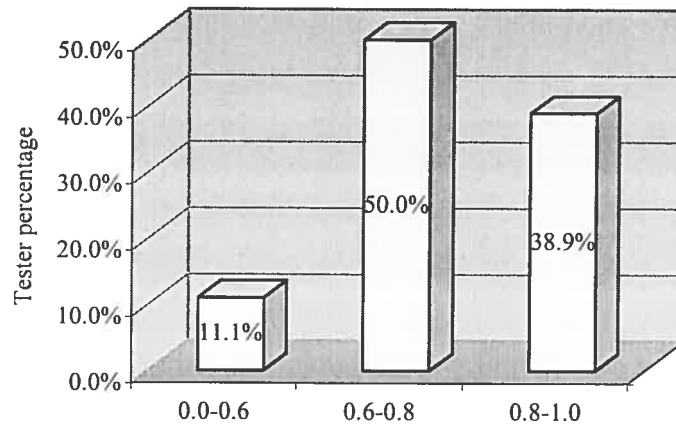


Figure 7.1 Tester percentage on three-point scale

Using the second method, we apply the large-sample test to invalidate the *Null Hypothesis* about the population mean. Hoping for a contradiction, we suppose that the mean user satisfaction rate is equal to 4 (the middle of our seven-point scale) after step 1. The test result is listed in Table 7.3. The low *P-value* allows us to reject the null hypothesis as being very unlikely, and therefore the mean satisfaction rate truly exceeds the average value 4. In other words, most people are satisfied with the method of Order-Based Similarity Measure.

Table 7.3 Result of Step 1

No. of Users	Mean	SD	z-value	P-value
36	4.58	1.20	2.90	0.01

Our next undesirable hypothesis is that it makes no difference whether we consider the recommendations of *PCFinder* before or after weight modification. But the test result shown in Table 7.4 allows us to reject this hypothesis and conclude that the method of weight modification was effective: as *P-value* is low, the difference between step 1 and step 2 is significant. This indicates that most people think that

weight modification can improve the performance of Order-Base Similarity Measure.

Table 7.4 Result of Step 1 vs Step 2

Step 1 vs Step 2	No. of Users	Mean	SD	z-value	P-value
Step 1	36	4.58	1.20	1.76	0.04
Step 2	36	5.07	1.13		

We use the same method as above to compare step 2 with step 3. We formulate the undesirable hypothesis that the recommended result shows no difference between before adaptation and after adaptation. The very low *P-value* indicated in Table 7.5 invalidates this hypothesis and shows that there is a significant difference between before and after applying the method of adaptation. This corroborates that most people think the method of adaptation can improve performance.

Table 7.5 Result of Step 2 vs Step 3

Step 2 vs Step 3	No. of Users	Mean	SD	z-value	P-value
Step 2	36	5.07	1.13	2.41	0.01
Step 3	35	5.74	1.22		

Now we turn our attention to the three cases outlined at the beginning of section 7.1. At first, we make the undesirable hypothesis that the recommended result shows no difference between basic configuration and advanced configuration. In table 7.6, the very low *P-value* invalidates this hypothesis and shows that there is a significant difference between basic configuration and advanced configuration. This illustrates that most people think that the recommended results with short-term constraints make them more satisfied than without short-term constraints.

Table 7.6 Result of Case 1 vs Case 2

Case 1 vs Case 2	No. of Users	Mean	SD	z-value	P-value
Case 1	36	4.58	1.20	1.87	0.03
Case 2	36	5.06	0.92		

Finally, we make the undesirable hypothesis that the recommended result shows no difference between users with or without long-term constraints. Again, the low *P-value* shown in Table 7.7 invalidates this hypothesis and indicates that there is a significant difference between users with long-term constraints and users without them. This illustrates that most people think that applying long-term constraints can produce more satisfactory results than not applying long-term constraints.

Table 7.7 Result of Case 1 vs Case 3

Case 1 vs Case 3	No. of Users	Mean	SD	z-value	P-value
Case 1	36	4.58	1.20	1.55	0.06
Case 3	36	5.13	1.23		

In order to visualize the mean user satisfaction rate for the three steps, namely before weight modification, after weight modification and after adaptation, the results are shown in Figure 7.2. The mean satisfaction rate before weight modification is 4.58. The mean satisfaction rate after weight modification is 5.07, which is higher than the rate before weight modification. The mean satisfaction rate after adaptation is 5.74, which is higher still than before adaptation. This comparison shows that the satisfaction rate after weight modification and adaptation is higher than before weight modification. Therefore, it demonstrates that the proposed methods of weight modification and adaptation can improve the performance of the Order-Based Similarity Measure.

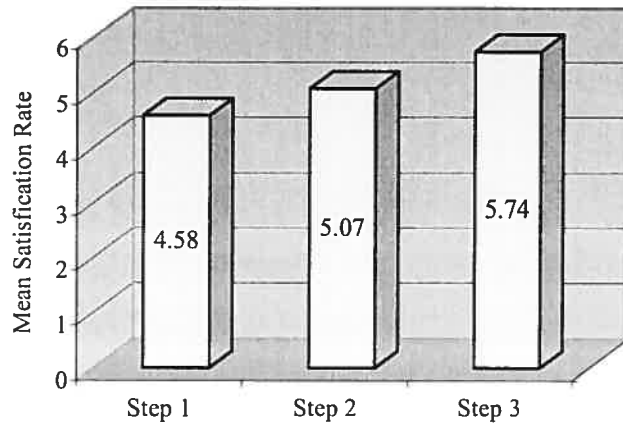


Figure 7.2 Comparison of the satisfaction rates for the three steps

In order to visualize the mean user satisfaction rate for the three cases, namely unregistered users using the basic configuration, unregistered users using the advanced configuration and registered users with constraints using the basic configuration, the results are shown in Figure 7.3. The mean satisfaction rate for case 1 is 4.58, the rate for case 2 is 5.06, and the rate for case 3 is 5.03. The mean satisfaction rate for case 2 is higher than for case 1. This demonstrates that the system can satisfy the user more effectively using short-term profiles than without them. The satisfaction rate for case 3 is also higher than for case 1. This indicates that the system can satisfy the user more effectively using long-term profiles than without them. The satisfaction rate in cases 2 and 3 is essentially the same, which gives indication that there is no significant difference between using short-term or long-term profiles.

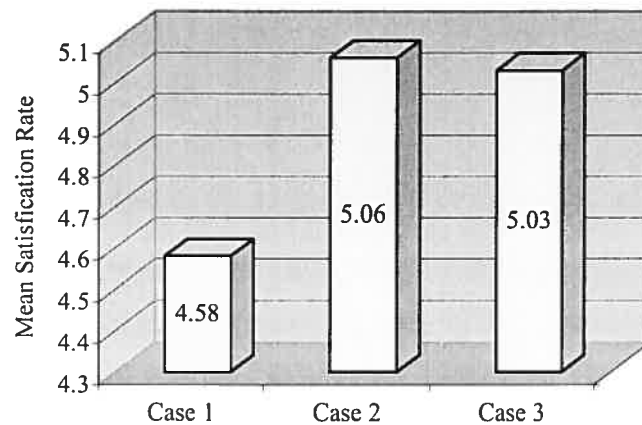


Figure 7.3 Comparison of the satisfaction rates for the three test cases

According to the test result, we make the following conclusion:

- Most people are satisfied with the method of Order-Based Similarity Measure.
- Most people think that weight modification can improve the performance of Order-Based Similarity Measure.
- Most people think that the method of adaptation can improve the performance.
- Most people think that the recommended results with short-term constraints make them more satisfied than without short-term constraints.
- Most people think that applying long-term constraints can produce more satisfactory results than not applying long-term constraints.

Conclusions and Future Work

In this thesis, we started by raising the issue of the lack of customer support and marketing analysis tools in electronic commerce applications on the Internet. Then we provided an overview of some major methods to solve this problem: Case-Based Reasoning, collaborative filtering, clustering techniques, etc. In particular, systems based on Case-Based Reasoning are a promising approach for the creation of agents that recommend products according to the specific requirements of customers.

In order to illustrate the methods we proposed in chapter 5, we constructed the architecture for an intelligent agent and implemented an online computer store. This store includes a case base of 150 cases and provides two different user-friendly interfaces, one assisting the consumer and the other assisting the market research staff.

PCFinder is an intelligent agent that uses a system based on CBR and collaborative filtering technologies to provide customer support in electronic commerce applications. First, it retrieves a set of broadly similar cases using the Order-Based Similarity Measure. Then, if the customer is not satisfied with the result, it can modify the weight of an attribute according to the customer's criteria. To increase the customer's satisfaction, *PCFinder* can also adapt the recommended result. In the retrieval step, the system applies short-term and long-term customer profiles to the similarity measure process in order to narrow down the number of appropriate cases and therefore accelerate the system's response time.

The main contributions of our work are:

- Proposing novel methodologies based on CBR for an e-commerce application. These include the dynamic Order-Based Similarity Measure, weight modification and adaptation. The dynamic Order-Based Similarity Measure does not require storage and can be reused, thus reducing maintenance activities and expenses. Weight modification allows the system to fulfill the customer's needs for the attributes most important to him. The adaptation module divides the product (or internal) attributes into independent, dependent and related attributes. In this module, the adaptation process varies with the type of attribute.
- Applying multiple technologies to product recommendation systems. The core of the system is an intelligent agent. CBR and collaborative filtering techniques were employed to make this agent more efficient and effective. Clustering analysis techniques were used to help the intelligent agent group customers according to their long-term profile; this allows the agent to later analyze the user profiles (external attributes) and provide product suggestions (internal attributes).
- Introducing a method for constructing a product recommendation system. We describe all aspects of the recommendation system: from architecture to methodologies and from applied technologies to implementations.
- Providing a graphic-building wizard based on clustering analysis of purchase records allowing the management staff to analyze market tendencies. Using records of past purchases, this wizard plots the relationship between items in the user profile (external attributes) and product attributes (internal attributes).

Despite our effort to build a helpful intelligent agent to assist the customer buying process, areas for improvement remain. As for future work, we intend to be engaged in more detailed studies. Several avenues for future work can be pursued, such as:

- Extending our work to a wider range of applications;
- Enabling *PCFinder* to communicate with other agents;
- Enabling *PCFinder* to extract semantic information.

PCFinder can potentially provide a large range of services to customers and increase their satisfaction with existing ones. This results in increased sales, making the company more profitable.

References

[Aamodt & Plaza, 1994] Aamodt, A. and Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches, *AI Communications*, Vol. 7(1). pp.39-59, 1994.

[Active Decisions] Active Decisions web site: <http://www.ActiveDecisions.com>

[Aguzzoli *et al.*, 2002] Stefano Aguzzoli, Paolo Avesani, and Paolo Massa. Collaborative Case-Based Recommender System, *6th European Conference (ECCBR 2002)*, pp. 460-474, Aberdeen, Scotland, UK, September 2002.

[Aïmeur & Vézeau, 2000] Aïmeur, E. and Vézeau M., Short-Term Profiling for a Case-Based Reasoning Recommendation System, *Eleventh European Conference on Machine Learning (ECML2000)*, pp. 1-12, May, Barcelona, Catalonia, Spain, 2000.

[Aïmeur & Xiao, 2003] Aïmeur, E. and Xiao, B., Personalizing Product Recommendation in E-Commerce, *The IFIP (International Federation for Information Processing) Conference on E-Commerce, E-Business & E-Government - I3E2003*, São Paulo, Guarujá, Brazil, September 21-24, 2003.

[Allamaraju *et al.*, 2001] Allamaraju, S., Ashri, R., Darby, C., Flenner, R., Karsjens, T., Kerzner, M., Krotov, A., Linde, A., MacIntosh, J., McGovern, J., Mirchandani, T., Plaster, B., Reamy, D., Dr P G Sarang, Writz, D., *Professional Java E-Commerce*, published by Wrox Press Ltd., UK, 2001.

[Amazon] Amazon web site: <http://www.amazon.com/>

[Bergmann *et al.*, 2002] Bergmann, R., Schmitt, S., Stahl, A., Intelligent customer support for product selection with case-based reasoning, *E-Commerce and Intelligent Methods. Studies in Fuzziness and Soft Computing*, Vol. 105, pp.322-341, Physica-Verlag, 2002.

[Berson *et al.*, 1999] Alex Berson, Stephen Smith & Kurt Thearling, *Building Data Mining Applications for CRM*, The McGraw-Hill companies, Inc., 1999.

[Bradshaw, 1997] Jeffrey Bradshaw, *Software Agents*, published by the AAAI Press/the MIT Press, 1997.

[Bridge, 2001] Derek Bridge, Product Recommendation Systems: A New Direction, R.Weber and C.G.von Wangenheim (eds.), the *Workshop Programme at the Fourth International Conference on Case-Based Reasoning*, pp.79-86, 2001.

[Bryan, 1997] Martin Bryan, *An Introduction to the Extensible Markup Language (XML)*, The SGML Centre, 1997. <http://www.personal.u-net.com/~sgml/xmlintro.htm>

[Burke, 2000] Burke, R., Knowledge-based Recommender Systems. In A. Kent (ed.), *Encyclopedia of Library and Information Systems*. Vol. 69, Supplement 32. New York: Marcel Dekker, 2000.

[Burke, 2002] Burke, R., Hybrid Recommender Systems: Survey and Experiments, *User Modeling and User-Adapted Interaction*. 12(4), pp.331-370, 2002.

[CoCoA] CoCoA web site: <http://www.karadar.com/>

[COGITO, 2001] COGITO web site:

<http://www.darmstadt.gmd.de/%7Eecogito/Description.htm>

[Cunningham *et al.*, 2001] Cunningham, P., Bergmann, R., Schmitt, S., Traphöner, R., Breen, S. and Smyth, B., Intelligent Support for Online Sales: The WEBSELL Experience, *the Fourth International Conference on Case-Based Reasoning (ICCBR 2001)*, pp. 87-93, Harbor Center in Vancouver, British Columbia, Canada. 31 July 2001.

[DEMOletter, 2001] DEMOletter Archives, "Success the Active Research Way: Guided Selling, Turning Queries Into Sales",
<http://www.demo.com/archives/demoletter/v5n11/3.html>, Nov. 5, 2001.

[Expedia] Expedia web site: <http://www.expedia.com>

[Firepond] Firepond web site: <http://www.firepond.com>

[Guttman *et al.*, 1997] Guttman, R., P. Maes, A. Chavez, and D. Dreilinger, Results from a Multi-Agent Electronic Marketplace Experiment, In Proceedings of *Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'97)*, Ronneby, Sweden, May 1997.

[Guttman *et al.*, 1998] Guttman, R. H., Moukas, A. G., Maes, P., Agent-Mediated Electronic Commerce: A Survey. *Knowledge Engineering Review*, Vol.13, No.2, 1998. Available at: <http://ecommerce.media.mit.edu/>

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

[Han & Kamber, 2001] Han, J. and Kamber, M., *Data Mining: Concepts and Techniques*, pp335-394, Morgan Kaufmann Publishers, 2001.

[Hayes & Cunningham, 2000] Hayes, C., Cunningham, P., Smart Radio: Building Music Radio on the Fly, presented at *Expert Systems 2000*, Cambridge, UK, December 2000.

[Hermans, 1997] Hermans B., Intelligenet Software Agent On the Internet: An Inventory of Currently Offered Functionality in the Information Society and a Prediction of (Near) Future Development, *FirstMonday-Peer-Reviewed Journal on the Internet*, 1997. http://www.firstmonday.dk/issues/issue2_3/ch_123

[IBM Education] IBM education web site:

http://www-900.ibm.com/developerWorks/cn/education/java/j-ejbfund/j-ejbfund_eng/j-ejbfund-2-2.shtml

[IBM Shopping] IBM shopping web site:

http://www-132.ibm.com/content/home/store_IBMPublicUSA/en_US/ready2buy.html

[Jango] Jango web site: <http://jango.excite.com>

[Jennings & Wooldridge, 1998] N. R. Jennings and M. J. Wooldridge, Applications of Intelligent Agents in *Agent Technology: Foundations, Applications, and Markets*, Springer Verlag, pp.3-28, 1998.

[Jola] Jola web site : <http://www.jola-info.de/>

[JSP] JSP web site: <http://java.sun.com/products/jsp/>

[Kolodner, 1993] Janet Kolodner, *Case-Based Reasoning*, published by Morgan Kaufmann Publishers, Inc., pp. 394, 1993.

[Krulwich, 1996] B. Krulwich, The BargainFinder agent: Comparison price shopping on the Internet. In *Agents, Bots, and other Internet Beasties*, SAMS.NET publishing, pp. 257-263, 1996.

[Kumar, 1992] V. Kumar, Algorithms for Constraint Satisfaction Problems: A Survey, *AI Magazine*, 13(1), pp. 32-44, 1992.

[Lang, 1995] Ken Lang, NewsWeeder: Learning to filter netnews. In *Machine Learning: Proceedings of the Twelfth International Conference*, Lake Tahoe, California, 1995. <http://citeseer.nj.nec.com/lang95newsweeder.html>

[Luck *et al.*, 2002] Michael Luck, Peter McBurney, Chris Preist and Christine Guilfoyle, *Agent Technology Roadmap*, published by AgentLink, October 2002. <http://www.AgentLink.org>

[Maes *et al.*, 1999] Pattie Maes, Robert H. Guttman, Alexandros G. Moukas, Agents that Buy and Sell: Transforming Commerce as we Know It, *the Communications of the ACM*, March 1999.

[Ma, 2001] Yanping Ma, *XMLFinder: an Intelligent Agent Based on CBR for E-Commerce*, Master thesis of Computer Science, at University of Montreal, 2001.

[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*, pp. 273-278, MIT, Cambridge, MA, June 2001.

[Mendenhall, 1983] William Mendenhall, *Introduction to probability and statistics*, sixth edition, pp. 302-306, PWS Publishers, 1983.

[Middleton *et al.*, 2001] Middleton, S.E., De Roure, D. C., and Shadbolt, N.R., Capturing Knowledge of User Preferences: ontologies on recommender systems, In Proceedings of the *First International Conference on Knowledge Capture (K-CAP 2001)*, Oct 2001, Victoria, B.C. Canada.

[MIT Media Lab] MIT Media Laboratory web site:

<http://agents.www.media.mit.edu/groups/agents/projects/>

[Moukas, 1997] A. Moukas, Amalthea: Information Filtering and Discovery using a Multiagent Evolving System, *Journal of Applied AI*, Vol. 11, No. 5, pp.437-457, 1997.

[Moukas *et al.*, 1998] Moukas, A., R. Guttman, and P. Maes., Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective, Proceedings of the *First International Conference on Electronic Commerce (ICEC'98)*, Seoul, Korea, April 1998.

[Netcraft] Netcraft web site: <http://www.netcraft.com/survey/>

[NewsWeeder] NewsWeeder web site:

<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/pleiades.html>

[Nwana, 1996] Hyacinth S. Nwana, Software Agents: An Overview, *Knowledge Engineering Review*, Vol. 11, No 3, pp. 1-40, Sept 1996.

[Pennock & Horvitz, 2000] D.M. Pennock and E. Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, 2000.

[Personalogic] Personalogic web site:

<http://www.personalogic.aol.com/go/gradschools/>

[Resnick & Varian, 1997] Resnick, P. and Varian, H. R., Recommender Systems, *Communications of the ACM*, 40(3), pp. 56-58, 1997.

[Ricci *et al.*, 2002] Francesco Ricci, Bora Arslan, Nader Mirzadeh and Adriano Venturini, ITR : A Case-Based Travel Advisory System, *6th European Conference (ECCBR 2002)*, Aberdeen, Scotland, UK, September 2002.

[Ripper *et al.*, 2000] R. Ripper, M. F. Fontoura, A. M. Neto, and C. J. Lucena, V-Market: A Framework for Agent e-Commerce Systems, *World Wide Web (WWW)*, Baltzer Science Publishers, 3(1), pp. 43-52, 2000.

[Russell & Norvig, 1995] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, (pp. 31), Prentice Hall, Upper Saddle River, N.J., 1995.

[Sarwar *et al.*, 2000] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl, Analysis of Recommendation Algorithms for E-Commerce, *ACM Conference on Electronic Commerce (EC'00)*, Minneapolis, Minnesota, October 17-20, 2000.

[Schafer *et al.*, 1999] Schafer, J. B., Konstan, J. and Riedl, J., Recommender Systems in E-Commerce, *the First ACM Conference on Electronic Commerce (EC'99)*, Denver, CO, pp. 158-166, 1999.

[Schroeder & Noy, 2001] Schroeder, M. and Noy, P. Multi-agent Visualization Based on Multivariate Data, *the 5th International Conference on Autonomous Agent*, pp.85-91, 2001.

[Shardanand & Maes, 1995] Shardanand, U., Maes, P., Social Information Filtering: Algorithms for Automating "Word of Mouth". In Proceedings of *CHI'95 Conference on Human Factors in Computing Systems*, Vol. 1, pp. 210-217, ACM, 1995.

[Smyth & Cotter, 1999] B. Smyth & P. Cotter, Surfing the Digital Wave: Generating Personalised Television Guides using Collaborative, Case-based Recommendation, Proceedings of *the 3rd International Conference on Case-based Reasoning*, Munich, Germany, 1999.

[Sony] Sony Style web site: <http://www.sonystyle.ca/>

[Stahl & Bergmann, 2000] Stahl, A. & Bergmann, R., Applying Recursive CBR for the Customization of Structured Products in an Electronic Shop. *5th European Workshop on Case-Based Reasoning*, pp. 297-308, Springer Verlag, 2000.

[Terveen & Hill, 2001] Terveen, L. and Hill, W., Human-Computer Collaboration in Recommender Systems, In: J. Carroll (ed.): *Human Computer Interaction in the New Millenium*, pp. 223-242, New York: Addison-Wesley, 2001.

[Turban *et al.*, 2002] Turban, E., King, D., Lee, J., Warkentin, M., Chung, H. M., *Electronic Commerce 2002: A Managerial Perspective*, published by Pearson Education Ltd., 2002.

[Vetter & Pitsch, 1999] Vetter, M. & Pitsch, S., Towards a Flexible Trading Process over the Internet, presented on the *AgentLink AMEC SIG* (Network of Excellence on

Agent-based Computing, Special Interest Group: Agent-mediated electronic commerce) meeting in Barcelona (Spain), pp.148-162, September 1999.

[w3schools] w3schools web site: http://www.w3schools.com/xml/xml_usedfor.asp

[Watson & Marir, 1994] Watson, I. & Marir, F., Case-Based Reasoning: A Review, published in *The Knowledge Engineering Review*, Vol.9 No. 4, 1994.

[Wilke & Bergmann, 1998] Wolfgang Wilke and Ralph Bergmann, Techniques and knowledge used for Adaptation during Case-Based Problem Solving, *IEA/AIE*, Vol. 2, pp.497-506, Springer-Verlag, Berlin Heidelberg New York, 1998.

[Xiao *et al.*, 2003] Bin Xiao, Esmâ Aïmeur, and José Manuel Fernandez, PCFinder: An Intelligent Product Recommendation Agent for e-Commerce, *the 2003 IEEE Conference on E-Commerce (CEC'03)*, Newport Beach, California, on June 24-27, 2003.

[XML 1.0] Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000. <http://www.w3.org/TR/REC-xml>