An E-shopping System with Different Negotiation Model

H. Ouchiyama, T. Yamazaki, and R. Huang
Faculty of Computer and Information Sciences,
Hosei University,
Koganei-shi, Tokyo 184-8584, Japan
{m00k0007, n00k1042, rhuang}@k.hosei.ac.jp

Abstract
This paper proposes an agent based e-shopping system that uses different negotiation models to match different users and generates different negotiation model objects for adaptation of different user preferences. Five different levels from simple to sophisticated of negotiation models are described. How different models are matched to different users is explained and how adaptation of different user preferences is shown in this paper. Finally, an e-shopping system architecture is presented and how the e-shopping system works is demonstrated.

1 Introduction
E-shopping is a hot topic that has been brought out with the rapid developments of the Internet and Web technologies [3, 4, 5]. It is undeniable that daily life has become convenient with it. People do not drive to a store, do not travel to oversea, they can purchase the goods they want. People do not physically open a shop, build a store, they can sell goods. Currently, most online shopping systems are still limited to provide customers online access to a list of products, take customers’ order and deal payments electronically online. Negotiations are not taken place among buyers, sellers, even wholesalers and suppliers. Although in traditional retail markets, prices and other aspects of a trading are often fixed, leaving no room for negotiation. With widely use of the Internet, buyers no longer have physical space constraints and can get as much as information about goods from different shops via search engines or directly visit online shops, the traditional fixed price retail markets have to be changed to adapt to the Internet era since more and more stronger competition among online accessible shops. Negotiation of price and other issues becomes more and more important in e-shopping markets since shopping is not just finding prices but maximizing profit or obtaining best value for money. Negotiation is a key component in e-shopping systems. In general, a human performs negotiations on a goods based on his knowledge or information about the goods, his shopping experience, and his reasoning model or thinking way. Different people have more or less different reasoning models due to different knowledge and experience, and different reasoning models result in different levels of negotiations. For 5 years old children, negotiations may be simple and straightforward while for 40 years old business men, negotiations between them would be much more sophisticated and complex. The negotiation models [7,8,9] used in e-shopping are supposed to be able to reflect such different levels like negotiations between real-world humans. With reasonable negotiation models, agents can do automated negotiations for users they represent [1,2,6].

2 Negotiation Models
Negotiation levels of human, in fact, are continuously evolving with increasing knowledge/experience with time goes. It is difficult to model all different levels. Although a several models from simple to sophisticated are listed in this section, our emphasis is on addressing the important concept of evolution of negotiation models.

• A constraint based negotiation
This model is very simple. A buyer makes a negotiation price depending upon how much the buyer has in his/her pocket. For example, although the 7-year-old boy would like to buy a toy that is priced at 3 dollars in a flea market, he has only 2. What he could do is just to make a counter offer (2 dollars) that is what he has. This is a one step negotiation that is rather simple and straightforward. It is a kind of strategy often adopted by younger children.

Figure 1. A constraint based negotiation
• A goal oriented negotiation
In this model, a buyer negotiates with a seller based on the seller’s proposed price and the buyer’s reservation price. The objective of negotiation is to make ultimate value of his money. At least, agreed price should be lower than or equal to his reservation price. The calculation of counter offer is based on a simple formula,

\[ P_{\text{buyer offer}} = R_{\text{buyer}} - (P_{\text{seller}} - R_{\text{buyer}})/2 \]

Where, \( R_{\text{buyer}} \) is the reservation price of the buyer. If the seller agrees on the proposed price by the buyer, the negotiation is successful, otherwise the process continues until the seller agrees on a price that is lower than or equals to the buyer’s reservation price.

\[ P_{\text{buyer offer}} = R_{\text{seller}}(1 + \beta) \]

Where, \( R_{\text{seller}} \) is an estimated reservation price of the seller. \( \beta < 0 \) means the buyer is risking, \( \beta = 0 \) means the buyer is conservative, and \( \beta = 0 \) means the buyer is neutral, that is the buyer is neither risking nor conservative.

• A counter party RP based negotiation
– original Bazaar model [11]
This model uses a similar alternating-offers way to make negotiations. What a difference is this model uses a learning-based strategy to decide a reasonable counter offer by learning to know a seller’s reservation price. Learning is proceeding by the Bayesian updating rule in the course of interactions with counter party during negotiations.

\[ P_{\text{buyer offer}} = R_{\text{seller}}\beta \]

For each negotiation model, it is good idea to define each of them as an abstract class. Let us take the enhanced extended Bazaar model as an example. The abstract class can be defined below in Figure 4.

• A third party information based negotiation
– extended Bazaar model [12]
This model is an extension version of the original Bazaar model with taking wholesalers or suppliers into considerations when a buyer learns to know a seller’s reservation price and makes a counter offer decision. In real-life negotiation, wholesalers, in fact, play quite an important role in negotiations between a buyer and a seller. During the negotiations between the seller and the buyer, the seller is possible to negotiate the wholesale price with the wholesaler regarding a same concerned item. If the seller and the wholesaler reach a new agreed wholesale price, the seller may leave more negotiation room for the buyer psychologically. On the other hand, if the buyer knows wholesale price averagely goes down, the buyer may stand more firmly in his negotiations with the seller. With this model, negotiation efficiency in term of negotiation rounds and quality in term of joint utility are improved.

• A common/public information based negotiation
– enhanced extended Bazaar model [12]
In both the original Bazaar model and the extended Bazaar model, initial probability distribution of given hypotheses on a belief is uniform. However, it is possible to give a better estimated or a certain solution to initiate the probability distribution if there is a certain amount of knowledge that can be obtained no matter how. As for negotiations in e-shopping, knowledge and information is incomplete, we have to use so-called “lose bounded” or “relaxing” game theory [10] based approaches that do not require strict complete knowledge. The “lose bounded” game theory based approach along with common knowledge and public available information is used to find a better initial probability distribution of given hypotheses instead of uniform distribution so as to shorten learning process in negotiations.

For each negotiation model, it is good idea to define each of them as an abstract class. Let us take the enhanced extended Bazaar model as an example. The abstract class can be defined below in Figure 4.
public abstract class Enhanced_Extended_Bazaar
{
    NP1 np1; NP2 np2; ...; NPM npm;
    UP1 up1; UP2 up2; ...; UPM upm;
    public void Enhanced_Extended_Bazaar
    (NP1 np1, NP2 np2, ...; NPM npm,
     UP1 up1, UP2 up2, ...; UPM upm)
    {
        this.np1 = np1; ......; this.npm = npm;
        this.up1 = up1; ......; this.upm = upm;
    }
    ...... // other constructors
    public void set_up1(U1 up1)
    {
        this.up1 = up1;
    }
    ...... // other set methods set_up2, ...
    public void set_upm(UPM upm)
    {
        this.upm = upm;
    }
    public double RP_fromPublicInfo()
    {
        ...... // implement
    }
    public double Nash_bargainSolution()
    {
        ...... // implement
    }
    public void init_probability_distribution()
    {
        ...... // implement
    }
    public void update_probability_distribution()
    {
        ...... // implement
    }
    abstract int evaluation_agree()
    abstract int evaluation_quit()
    abstract double fitness_value()
    abstract double max_fitness()
    ...... // implement
    abstract double offer_solution()
    ...... // other methods
}

Figure 4. Enhanced_Extended_Bazaar abstract class

Where, np1, np2, ..., npm are a set of non user preference variables. up1, up2, ..., upm are a set of user preference variables. Similarly, other negotiation models can be defined.

3 Matching a User to a Negotiation Model

Five different levels of negotiation models have been described above. In fact, human negotiations in real life have much more levels. Facing so many different negotiation models, how to select one for an individual user (a buyer or a seller) becomes an important issue. Although any users can select a negotiation model manually by themselves, there is a mechanism to match a user to a negotiation model in our e-shopping system. In order to achieve this, at first, it is necessary to acquire the user’s characteristics or types and some personal information. Two ways to obtain such information, one straight way is to provide a user interface for the user to input necessary information. An alternative way is to trace the user online behaviors or activities and to summarize them into useful information. Secondly, it is to use relevant information for matching the user to a negotiation model by a mapping function as shown in Figure 5.

![Figure 5. A mapping function f for matching a user to a negotiation model](image)

After deciding a negotiation model for a user, for example, the enhanced extended Bazaar model, the next important step is how to take a user’s preferences into considerations. Some users may negotiate over a single issue like price, quality, quantity, or maker while some users may like to negotiate over a combination of some of the above single issues. It is important for a selected negotiation model to adapt to user’s preferences. To enable such adaptation, we have defined several subclasses of a negotiation model class, such as Price_Negotiation, Quality_Negotiation, Quantity_Negotiation, Maker_Negotiation, etc. Below in Figure 5, shows an example of subclass that is derived from the super class Enhanced_Extended_Bazaar.

4 Adaptation of a Negotiation Model to User Preferences

It is natural that people unconsciously take their personal preferences into considerations when they do shopping, individuals have different preferences, and their preferences may change from time to time. Someone may prefer a certain brand of products and don’t care about price while someone may be strict on looking for lower price. Someone likes to take a risk while someone is conservative. Someone likes to get bargaining by buying a quantity of products while someone would not like to do so. Someone likes red color in winter but light blue color in summer. How to make a negotiation model works well
for different users? Adaptation to user preferences is a critical point.

```java
public class Price_Negotiation extends Enhanced_Extended_Bazaar {
    ......
    public void Price_Negotiation(NP1 np1, ..., NPm npm,
        UP1 up1, ..., UPn upn) {
        super(np1, ..., npm, up1, ..., upn);
        ......
        ...... // other constructors
    int evaluation_agree(){
        ...... // implement
    }
    int evaluation_quit(){
        ...... // implement
    }
    double fitness_value(){
        ...... // implementat
    }
    double offer_solution(){
        ...... // implement
    }
    .other methods
}
```

Figure 6. Price_negotiation class

**4.1 User Preference Setting Interface**

To achieve the adaptation of user preferences, the first step is to acquire user preferences. As for shopping, user preferences may be divided into two categories: general preferences and specific preferences. The former is of generality and is a category of preferences that all users are likely to specify explicitly or implicitly such as price, quality, quantity, maker and what issue to be negotiated. Figure 7 shows an example of a user inputting his general preferences.

After setting user general/specific preference, the list of the goods set up by user preference comes out. The user can buy his favorite goods. User inputs what goods will be purchased and negotiated. Or he can also acquire the detailed information on the goods. Figure 8 shows an example of goods list after setting user general and specific preference.

Figure 8. Two examples of user specific preference setting interface

After deciding the goods to purchase and the user wants to negotiate, he will set “negotiation parameter”. The user inputs required information for negotiating, such as reservation price, suggestion price, and what issues he will negotiate. And he also selects the importance of the goods from three steps of “Goods Importance”. This is very helpful to matching different negotiation models. Figure 10 shows an example of negotiation model setting interface.

Figure 9. An example of good list shown to a user

The latter is a category of preferences that is specifically concerned with a certain type of products. Preference items vary with different kind of products. For computer related products like notebook PC, users may specify color, size, weight, CPU, and memory, while for automobile related products like car, users may specify number of door, capacity, engine, interior, and other optional parts. Figure 8 shows two examples of users setting specific preferences concerned two different kinds of products.

Figure 7. A user general preference setting interface
Of course a straightforward way of acquiring user preferences is to provide interfaces as shown in Figure 7, 8, 9 and 10 for users to input their preferences interactively. An alternative approach is to employ an agent to automatically look up user’s profile, trace user’s shopping behavior on Web and find out user’s personality so as to result user’s preferences. This approach is more intelligent involved but requires Web technology, data mining techniques, and sophisticated mapping functions to map acquired information to user’s preferences.

### 4.2 Generating Negotiation Model Object

When user preferences are acquired and then it is necessary to translate them into user preference parameters. The user preference parameters are stored in a vector \( \{u_{p_1}, u_{p_2}, \ldots, u_{p_n}\} \). Each element in vector is an object of a user preference class \( (U_{P_1}, U_{P_2}, \ldots, U_{P_n}) \). Acquired user preferences can be passed to a negotiation model object when the object is created or by a set of user preference setting methods, \( \text{set}_{u_{p_1}}(), \ldots, \text{and } \text{set}_{u_{p_n}}() \). If a user prefers to negotiate product price, the price negotiation model object is created.

```java
model_1 = new Price_Negotiation(..., u_{p_1}, u_{p_2}, ..., u_{p_n});
```

The instance object, \( model_1 \), of the class, Enhanced_Extended_Bazaar, is thus generated by calling a constructor method and the acquired user preference parameters are passed to the object as the constructor method’s arguments. When a changing user preference event is caught, the changed user preference value, for example, \( u_{p_1} \), is reset to \( model_1 \) object by setting method.

```java
model_1.set_{u_{p_1}}(u_{p_1});
```

Similarly, \( u_{p_2}, u_{p_1}, \ldots, \) and \( u_{p_n} \) can be reset to the \( model_1 \) in the same way when it is necessary.

```java
model_1.set_{u_{p_2}}(u_{p_2});
```

```java
......
model_1.set_{u_{p_n}}(u_{p_n});
```

For a different user, a different instance object with different user preference parameters can be generated. Users’ preferences can be thus reflected in the generated instance model object and events of updating user preferences can be caught and get response from the negotiation object. Negotiation is going to adapt to a newly updated situation gradually. Although it takes time to adjust, it certainly can adapt to updated user preferences through Bayesian learning method.

### 5 How an E-shopping System Works

Our e-Shopping System with the negotiation models embedded in is presented in Figure 11 and Figure 12. This system consists of Agent Server with Database connected and User Agent. User Agent are further divided into Buyer Agent and Seller Agent. They function as an agent manager, a buyer agent, and a seller agent, respectively. Since there are some common features and as well as settings, so an abstract class named “ShoppingAgent” that includes some abstract methods is defined and the AgentServer class and UserAgent class are extended from the ShoppingAgent class. BuyerAgent and SellerAgent are extended from the UserAgent.
Although the BuyerAgent and the SellerAgent are fundamentally same, there are still differences in negotiation model, data/information processing, and management. It is wise to differentiate them as two different classes that are extended from a same superclass. For example, both seller and buyer have a *Connection Object* for connecting to the Agent Server, and can receive goods information, in addition the seller agent can manage information but the buyer agent can not. The Agent Server holds objects of users and goods in its database, accepts agent registration from the buyer/seller agents, and manages those them. “Users Information” (object) consists of User ID, password, personal data (such as full name, e-mail address, and so on), purchase history, and User Preferences. When negotiations between a seller agent and a buyer agent start, mutual communications are conducted and managed by the Relay method in the Agent Server. For example, the agent server relays the negotiation parameters from a buyer agent to a seller agent and the seller agent makes a decision like price offer based on its negotiation model. To send and process negotiation data, an object of the class, “Shopping Agent Message”, is created. This object contains four kinds of data: Command, Sender_ID/Agent_Name, Receiver_ID/Agent_Name, and Parameter(s). When a buyer user makes an offer to a seller user, the buyer agent that represents the buyeruser creates “Shopping Agent Message” object as shown in Figure.13 and sends the object to the Agent Server.

**Figure. 12.** Inheritance hierarchy of Shopping Agents

In sum, the e-shopping system works as described below. First the agent administrator starts Agent Server and Seller Agent. Any seller should log in to seller agent via the Internet and put the goods information before the Buyer Agent is started. Then agent administrator starts the Buyer Agent. If a buyer user accesses to the Buyer Agent for the first time, the buyer must input user ID, password, and personal data to register with the Buyer Agent. This information is stored in the Agent Server Database. Next the buyer logs in to the Buyer Agent by inputting his User ID and password. To check whether the ID and password are correct, the Buyer Agent sends data to the Agent Server. If the data are matched, the Agent Server sends goods information to the Buyer Agent and the Buyer Agent shows it to the buyer user. The buyer user can set user preferences and specific preferences to search for required or preferred goods. The buyer user chooses goods to buy and decides whether to carry out price negotiation with the seller user or not. If the buyer user does not want to proceed to price negotiation, the window, which chooses the payment method etc., appears. If the buyer user wants to proceed to price negotiation, the buyer user must input buyer’s suggested price, reservation price, and what negotiation models he wants to use. The Buyer Agent tries to connect to the Seller Agent via the Agent Server. If he does not select a negotiation model, the system automatically matches the user to an appropriate negotiation model by the mapping function. Then negotiation between a seller and a buyer starts. If the negotiation

**Figure. 13.** Contents of “Shopping Agent Message” object

<table>
<thead>
<tr>
<th>Command</th>
<th>“RPrice”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender_ID/Agent_Name</td>
<td>Buyer / Buyer Agent xx</td>
</tr>
<tr>
<td>Receiver_ID/Agent_Name</td>
<td>Seller B / Seller Agent yy</td>
</tr>
<tr>
<td>Parameter(s)</td>
<td>Goods ID, Quantity, RPrice</td>
</tr>
</tbody>
</table>

Agent Server checks whether the Receiver_ID in the sent message is “Agent Server” or not. If so, Agent Server processes the command contained in the message. If not so, Agent Server relays the message to the receiver agent. The receiver (seller) agent processes the command contained in the message.
is successful, the buyer user can buy goods at the agreed price. After the shopping, the purchase history of the buyer is stored to the Database in the Agent Server.

6 Conclusions and Future Work

Negotiation is an important component in an e-shopping system. Like human negotiations in conventional business or trading, different users negotiate on different levels based on their experience and knowledge and they naturally apply their preferences implicitly to decision making in negotiations. To emulate negotiations in real life, negotiation modeling in an e-shopping system is mainly focused on matching a user to a developed negotiation model and adapting a negotiation model to a user’s preferences. In this paper, a mapping function is proposed to map a user to a negotiation model and different negotiation model objects can be generated to adapt different user preferences. Finally, an architecture and working principle of negotiation agents embedded e-shopping system is presented and demonstrated. There is still much remaining work for improving our system. Our future work will be focused on

- developing a matching agent instead of mapping function for automatically matching a user to a negotiation model,
- developing an intelligent system for evolving negotiation model with user's increasing knowledge and experience, and
- making our e-shopping system into practical use.

References