A RESTAURANT FINDER USING BELIEF-DESIRE-INTENTION AGENT MODEL AND JAVA TECHNOLOGY

Dongqing Lin
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202

Thomas P. Wiggen
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
(701) 777-3477
wiggen@cs.und.edu

Chang-Hyun Jo
Department of Computer Science
California State University, Fullerton
Fullerton, CA 92834
(714) 278-7255
jo@ecs.fullerton.edu

Contact Point:

Chang-Hyun Jo
Associate Professor
Department of Computer Science
California State University Fullerton
Fullerton, CA 92834-6870
(714) 278-7255
jo@ecs.fullerton.edu

A RESTAURANT FINDER USING BELIEF-DESIRE-INTENTION AGENT MODEL AND JAVA TECHNOLOGY

Dongqing Lin
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202

Thomas P. Wiggen
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
(701) 777-3477
wiggen@cs.und.edu

Chang-Hyun Jo
Department of Computer Science
California State University, Fullerton
Fullerton, CA 92834
(714) 278-7255
jo@ecs.fullerton.edu

ABSTRACT

It has been becoming more and more important to design systems capable of performing high-level management and control tasks in interactive dynamic environments. At the same time, it is difficult to develop and maintain such complex systems with our traditional software techniques. The agent-oriented/based systems, rooted in a different view of computational entities, offer prospects for a qualitative change in this aspect. In this work, we adopt the basic architecture of a Belief-Desire-Intention (BDI) agent model and develop a more intelligent and dynamic searching model for agent programming. In the BDI model, the Desire is the goal to achieve or event to handle, the Intention is a set of plans to realize the predefined goal or react to a specific situation, and the Belief is the knowledge about the agent itself and the varying environment. Applying BDI concepts, we will build an experimental framework in our Restaurant Finder system. The belief, desire and intention are initially established as separate classes as in the object-oriented analysis and design, then these classes will be integrated as basic units for a BDI agent. The agent will possess learning behavior based on the user's feedback and the principle of inferring preferences. In addition, the agent will also update its knowledge dynamically based on analysis of user's interaction with the system. The dramatic increase in the use and availability of mobile devices has resulted in the ability to access information anytime and anywhere. Our Restaurant Finder system is designed for use on mobile phones. This system has been constructed and implemented based on a three-tier architecture. The client tier is a J2ME (Java^{TM} 2 Micro Edition) emulator for the primary interface. In the middle tier, the Apache Tomcat Server will be installed to process the client's request, update the system database, and send back the recommendation. The information tier is a database system using the IBM Cloudscape database. The Restaurant Finder system will demonstrate an example of BDI agent programming and the J2ME Mobile Information Device Profile (MIDP) client application design.

Keywords

Belief-Desire-Intention (BDI) Agent-Based Programming

1. INTRODUCTION

1.1 The Problem

Conventional computer software and systems have been designed and built for dealing with precise and complete information. In recent years, it's often required that new systems should be capable of working with a complex and uncertain world. These systems will need to process imperfect and limited information from a substantially changing environment. In addition, it is often required that the existing systems will have to be changed and modified frequently to improve their capability and performance. Efficient and flexible computer architectures and languages that reduce the complexity and time for system specification and modification have been developed to meet these demands [Georgeff et al. 98]. Among them, the agent-based modeling techniques provide effective architecture to modularize the sophisticated systems and solutions to handle the interaction between the systems and the environment dynamically.

Agent computing is based on agents. An agent is a concurrent, autonomous, intelligent, and self-contained object. Here the self-containing has two meanings. First, an agent has a definite goal. Secondly, an agent defines pertinent behavior to achieve its goal based on the current environment [Jo 01]. The Belief, Desire, and Intention construct the essential part of the state which best describes the systems and the environment. The Belief represents knowledge of the world, i.e. the agent itself and the varying local environment. It can be the value of a variable, a relational database, or symbolic expressions in predicate calculus. The Desire is the goal to achieve or event to handle, which can be the value of variable, a record structure, or a symbolic expression in some logic. The important point is that the Desire represents the desired end state. By defining states, the corresponding behavior of the agent can be determined step by step to achieve the goal. Applying this agent Desire concept will lead to the desire-oriented computing instead of conventional task-oriented computing. The desire includes relevant tasks to be completed. The desire-oriented computing focus on what to do instead of how to do as defined in task-oriented computing. The Intention represents the third necessary component of the state. It is a set of plans to realize the predefined goals or react to a specific situation. Computationally, Intention may be a set of executing threads in a process [Georgeff et al. 98].

The agent model based on Belief-Desire-Intention (BDI) has been proved as a powerful computing technique [Bratman 87]. One interesting BDI agent system called JACK has extended the Java programming language in agent modeling [JACK 99]. JACK has Class, Interface, Method, Syntactic and Semantic extensions of Java implemented as Java plug-ins to support an agent-oriented development environment. JACK also uses the Database class to define and manipulate its beliefs. Another Mobile BDI Agent Toolkit applies Java packages to provide a runtime environment for BDI model programming [Busetta et al. 98]. An application of BDI agent modeling, Agent-based Stock Trader shows how to design and implement the Belief-Desire-Intention concepts using Java classes and Microsoft Access database system [Feng & Jo 02]. Meanwhile new concepts for an agent-based programming language, APL, have recently been proposed and developed. One APL prototype is to translate the APL source into the Java source codes, which can be run on the Java Virtual Machine [Jo & Arnold 02].

Applying BDI concepts, we will build an experimental framework for our Restaurant Finder system. However, until now there have been no proper programming languages to well support agent programming naturally. So in this work, we try to enhance the capability of the object-oriented language, Java, to build this agent system. The belief, desire and intention are initially set up as separate object classes, then these classes will be integrated as basic units for a BDI agent. The dynamic mapping from the specific belief to an Intention plan will be realized by manipulating the relational database. In addition, the agent will possess learning behavior based on the user's feedback and the principle of inferring preferences. The agent will also update its knowledge dynamically to make it more intelligent by analyzing the user's interaction with the system.

1.2 The System

The dramatic increase in the use and availability of mobile devices has resulted in the ability to access information anytime and anywhere. Our Restaurant Finder system is built for use on mobile phones. A three-tier architecture design is applied in this system. The client tier is a Sun MIDP-device emulator that acts as the J2ME client that receives content in the plain-text format. J2ME^{TM} (Java^{TM} 2 Micro Edition) is Sun's newest Java platform for developing applications for various consumer devices, such as set-top boxes, embedded systems, mobile phones and cell pagers. MIDP (Mobile Information Device Profile) is a set of APIs that allows developers to handle mobile-device-specific issues, such as creating user interfaces, storing information locally and networking. In the middle tier, the Apache Tomcat Server will be installed to process the client's selections, update the system database, and send back the recommendation. This server is the Java servlet container to handle the user's Get/Post request and deliver a restaurant candidates screen or a restaurant recommendation screen to the user. The information tier is a database system using the Cloudscape relational database, which is a pure-Java database management system from the IBM Cloudscape [Deitel 02]. The Restaurant Finder system will demonstrate an example of BDI agent programming and the J2ME MIDP-device client application design.

1.3 Outline of the Document

In Section 2, we will describe the basic architecture and design of this BDI-agent based Restaurant Finder system. In Section 3, we will explain how to implement the BDI-agent concepts using Java technology and relational database in this system. An application example for testing the system will be shown in Section 4. The conclusions and ideas for future work are listed in Section 5.

2. CONCEPT AND DESIGN

2.1 BDI Agent Model

The first-order intentional system, which has beliefs and desires, but no beliefs and desires about beliefs and desires, is the basis for our intelligent Restaurant Finder system. In this BDI agent model, we have adopted human-like mental attitudes of Belief, Desire, and Intention as the blocks representing the information, motivation and deliberation of the agent. This kind of modeling mechanism has been argued as rational and flexible for building an intelligent goal-driven agent [Rao et al. 95].

One purpose of this work is to show that the BDI agent model can be used to build a system that can learn and adapt its behavior according to accumulated experiences and changing environment. The user's choices, restaurant information, and local environments will be recorded as basic elements of Belief. Initially the criteria will be set up for choosing a most appropriate restaurant based on the user's preference and current situation. The plans, i.e., Intention, will be called to meet the criteria and achieve the goal. When a user searches the restaurants, various plans that reflect the current and local information will be utilized to prepare a list of restaurant candidates at run-time. Then the system can make comparisons and give the user a final recommendation based on this itemized group of restaurants. Here the agent model has the exact Desire to achieve. It will let users select the restaurant quickly and satisfactorily, and will re-build the information database for the agent model each time the user makes a choice. This Restaurant Finder application also demonstrates the computability of the BDI model in constructing a real-world system. The concepts of Belief, Desire, and Intention can be effectively designed and implemented in a way similar to the normal practice applied in unified process of software engineering nowadays [Einhorn and Jo 02, Jo 01].

2.2 Architecture Design

The three-tier design has been applied to the Restaurant Finder as in Figure 1.

In the Restaurant Finder system, the first layer is the client part that is the user interface. The J2ME MIDP Emulator will be used to simulate the mobile phone screen. The second layer is the Web server part, which is the servlet engine for the Restaurant Finder system. The third layer is the information storage, which is also a server for the relational database management system. The interface will direct the user request to be processed at the corresponding BDI agent built using the Java Servlet technology. The response, i.e., the restaurant recommendation from the system, or the initial restaurant candidate list from user profile, will be shown through the same user interface too. The BDI agent-based system will retrieve, maintain, and update the database system. The main contents of the database are user profile, restaurant features, and local environment. The system has been installed on the Apache Tomcat server. The Java-based IBM Cloudscape database will be used to act as the database system for the agent to rely on.

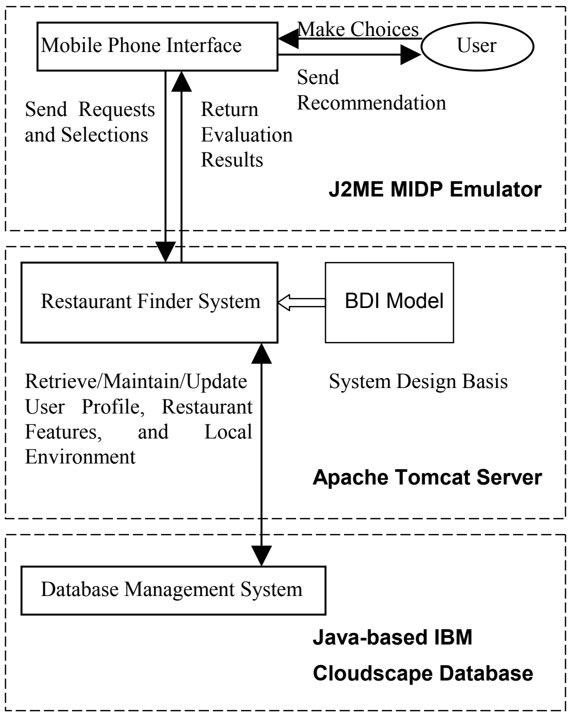

Mobile Phone Interface
Make Choices
User
Send Recommendation
Send Requests and Selections
Return Evaluation Results
J2ME MIDP Emulator
Restaurant Finder System
BDI Model
Retrieve/Maintain/Update User Profile, Restaurant Features, and Local Environment
System Design Basis
Apache Tomcat Server
Database Management System
Java-based IBM Cloudscape Database

Figure 1. Three-tier Architecture of The Restaurant Finder System.

Each time the user logs in, the information about the local environment and restaurants will be provided according to the location and the time that the user uses the mobile phone. The Restaurant Finder system will first propose to the user a predefined candidate list of the restaurants based on current conditions and user's preference recorded in the database. The user then can either choose a restaurant from the list or make new selections from a menu to be shown on the mobile phone screen. The system will make an optimum choice to meet the user's goal by selecting the most appropriate restaurant from the database. Meanwhile the system will also update the database system to reflect the user's new preferences based on his/her current choices. This can also be used to update the criteria in the database system to be used as the plans for building future restaurant candidate list. In this way, the system is able to remember user's preferences and provide a more recent candidate list next time the user logs in. Thus the system is designed with a dynamic learning ability in the run-time environment by manipulating the database system. The basic reasoning mechanism is to assign a specific weight value to each element in the database and calculate the total amount for each possible selection of a restaurant. The weight values are also subject to change after each selection made by the user.

3. IMPLEMENTATION OF THE SYSTEM

3.1 BDI Agent-based Programming

Agent-based programming provides a new paradigm in building intelligent agent systems. Instead of writing complicated programs using object-oriented or even structured modeling methodology, agent-based programming extracts goal-driven autonomous, perceptive and cooperative agent concepts from the problem domain. However there have been no proper programming languages that can be directly used to construct agents. So in our work, we take advantage of the object-oriented nature of the Java language and program our agents based on object concepts. Although Java does not support directly BDI agents in our case, Java has suitable class structures that can be used to build the basic units, like Belief, Desire, and Intention in an agent model. For the time being, a relational database system is chosen to support the Java BDI-agent programming. Although the class structures of BDI concepts cannot be changed at run-time, the information in the database can be updated dynamically according to the changing environment. The information system will be re-built every time a new restaurant candidate list is generated for the user using Java Servlet technology.

3.2 Core of BDI Agent

As we described in Chapter 2, the relational database management system will be used to establish the database system for the BDI agent. The learning and reasoning ability of the BDI agent will be realized dynamically with the aid of this database system. Using Structured Query Language (SQL), the Belief elements will be stored, retrieved, and updated in several tables of the database system. Similarly, criteria used to suggest restaurants will also be stored and updated in an Intention mapping table. The restaurant candidate list will be constructed by retrieving these restaurant items to meet user's requests for different environments.

For this application, the main items about the restaurant feature are type, price, location and parking which are established in the Belief table of the database. In the database, the above-mentioned items are the basis for the Belief concepts. Each element for these items will be assigned a weight value from 1 to 10, representing the quality or grade for each element. The higher value means a restaurant closely satisfies user's preference according to this item. These values will be updated each time the user makes different choices and a final decision. Recommendations will be made by the system according to the comparison of total weight values of available restaurants in terms of these data items. The system will advise the user dynamically by changing the weight values each time user makes new choices. Even in the user interface part that is the mobile phone screen, the items shown on the menu each time will also be different based on the current situation and the user's former selections. The restaurant candidate list will be built and updated each time the user starts the system. Based on previous information, an initial candidate list will be shown on the mobile screen when the user starts the system. This will accelerate the process to let the user make a decision in an acceptable or short time. According to the current Belief items and the user's preference, this list will be different and customized each time the user logs in. After that, a dynamic menu will be shown on the screen for the user to make new choices if the user has a different or new idea than the recommended restaurants in the candidate list. By calculating the total weight values of all the concerned Belief elements, a new optimum restaurant candidate list will be provided to the user.

3.3 Structure of BDI Agent

For this Restaurant Finder system, the basic data structure for Belief, Desire and Intention have been developed based on the relational database table. The Desire is clear, which is to suggest an appropriate restaurant to satisfy user's requirement. The contents for the Belief are maintained in two tables as described below.

Table 1. Database Tables for Belief

Table Name	Content
Restaurant	Information about available restaurants around.
Profile	User's profile for selecting restaurants.

The Restaurant table represents the current local situation in which the user starts the Finder system. They will be generated dynamically and will simulate the real environment for the restaurants in the surrounding area. Each time the user logs in, the selection process of restaurant will be performed according to varying conditions. The detail of the Restaurant table is as follows:

Table 2. Restaurant

ID
Name
Type
Price
Location
Parking
Address

The core part of the Belief is the user Profile table, which provides permanent storage for the user's history list of selected restaurants. Initially, an empty table only with meta-data schema is designed for the first time the user starts the Finder system. Then the information about the user's selections will be recorded in the table after the user successfully makes decisions. The schema of the Profile table is the same as that of the Restaurant table.

Table 3 Profile

ID
Name
Type
Price
Location
Parking
Address

The difficult but interesting part is how to implement the Intention dynamically. The user Profile table is the basis for making reasonable recommendation to the user. Each time the user logs in, the system will first analyze the user's previous selections with available local restaurants. Then the system will provide the user a restaurant candidate list that reflects the user's previous selections. If the user has a new idea, the system will also be able to provide a customized shortcut menu for the user to choose. The analysis is based on the frequency of restaurants appearing in the Profile table. Here a Java two-dimensional array will be used to construct such a candidate list first. All the information about a specific selection can be included in this array easily. The array size will be determined dynamically based on the Profile table information when the user logs in. After the user selects a restaurant, all the pertinent information will be added to the user Profile table for future reference. Note that the Intention is about the several plans to suggest different possible solutions to the user. The plans are chosen according to user's interests that may vary every time using the Finder system. These interests are itemized as different criteria in building plans. These plans can also be combined to make an optimal suggestion to the user. This process reflects a dynamic mapping from Belief to Intention with the help of database system. An Intention mapping table is built to simulate the dynamic mapping in this application.

For one instance, Intention class is implemented to provide information for use by the BDI agent in this application. Based on user's selection, the pertinent restaurant type, price, location and parking will be converted into corresponding integer numbers from 1 to 10. The agent will sum these numbers up for each restaurant in the local area. The restaurant with highest total weight value will be shown in the mobile phone interface. The above mentioned process implemented by the Intention will be controlled dynamically by using an Intention mapping table as shown in Table 4. In the IntentionMap table, iName, iType, iPrice, iLocation and iParking are the base values to be used by the Intention class. The iNum is a control number to define which set of data in the IntentionMap table will be applied this time. By giving different value of iNum, the Intention class will be able to determine which plan is to apply for the current case. In such a way, the BDI agent would probably provide different recommendation to the user even for the same local environment.

Table 4. IntentionMap

iNum
iName
iType
iPrice
iLocation
iParking

Modular design can be easily applied to the BDI agent-based system. First the Belief, Desire and Intention will be built based on Java classes. Then these pre-defined structures will be assembled to build an agent. The basic structure of BDI agent-based Restaurant Finder system is shown in Figure 2.

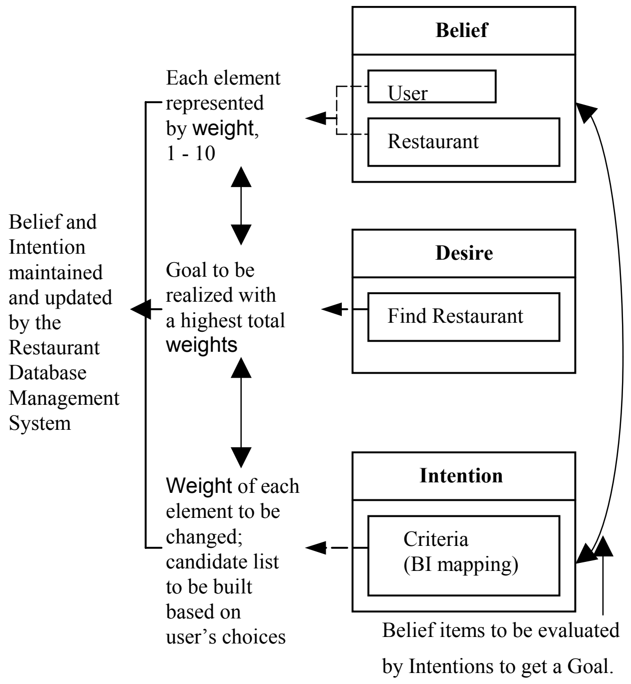

Belief
User
Restaurant
Each element represented by weight, 1 - 10
Desire
Find Restaurant
Goal to be realized with a highest total weights
Belief and Intention maintained and updated by the Restaurant Database Management System
Intention
Criteria (BI mapping)
Weight of each element to be changed; candidate list to be built based on user's choices
Belief items to be evaluated by Intentions to get a Goal.

Figure 2. Basic Structure of BDI Agent-based Restaurant Finder

4. EXPERIMENTAL RESULTS

4.1 Simulated Local Environment

The restaurant Finder system has been tested by applying a simulated local environment. It is assumed that the user will start the system in an area with seven restaurants. The pertinent information about these restaurants is listed in Table 5.

Table 5. Restaurant Information for Testing.

Name	Type	Price	Location	Parking
Applebee	American	High	Nearby	Difficult
Burger King	Fast Food	Very Low	Close	Easy
China Garden	Chinese	Medium	Far	Easy
Guadalajara	Mexican	Medium	Very Close	Very Difficult
Olive Garden	Italian	High	Very Far	Very Difficult
Pizza Hut	Italian	Medium	Close	Easy
Red Lobster	American	Very High	Far	Difficult

The addresses of the restaurants are not listed, but they are stored in the Restaurant table in the database. They represent a typical scenario that the Restaurant Finder system should handle. In processing the restaurant information, each data item will be assigned a weight value according to the Intention plan selected by the system and user's choices. Then the total weight values will be compared by the system and a most appropriate restaurant will be provided to the user. In the meantime, the system will record the user's choices and update the user's Profile table.

4.2 Example for the First-time User

Figure 3 shows the Welcome screen of the Restaurant Finder system. When the Select button is clicked, it will direct an initial message to the Candidate screen as shown in Figure 4.

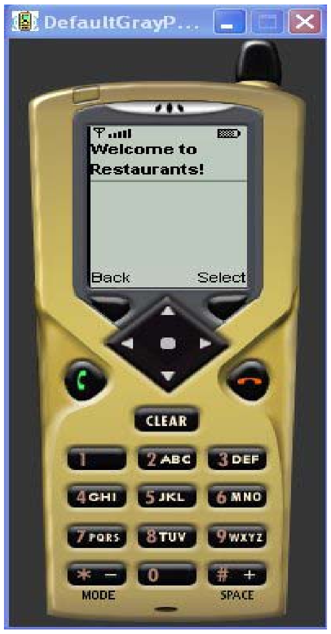
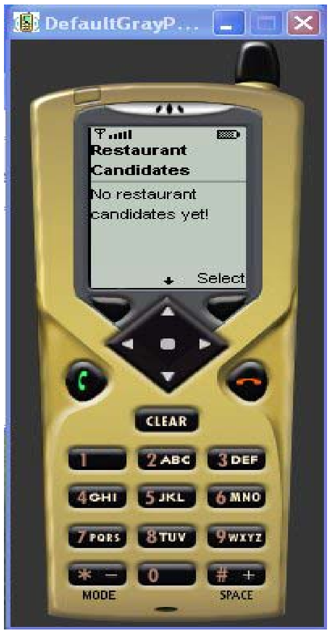


Figure 3. Welcome Screen.



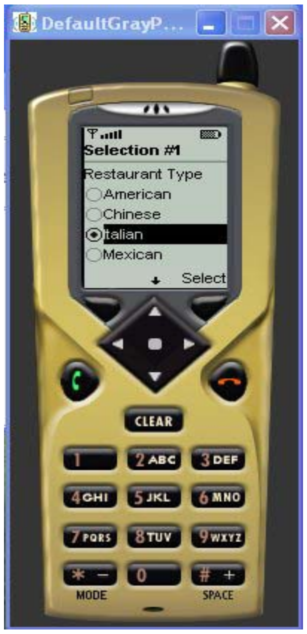
Restaurant Candidates

No restaurant candidates yet!

Select

Figure 4. Candidate Screen.

As this is the first time to use the system, there is not any record in the user Profile table. The screen shows a message “No restaurant candidates yet!”. The user needs to input his/her preferences and let the system recommend a restaurant. A series of four screens must be gone through as shown in Figure 5, 6, 7, and 8.



Selection #1

Restaurant Type

American

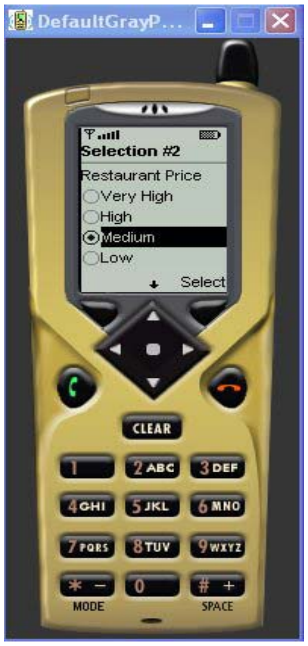
Chinese

[✓ Italian]

Mexican

Select

Figure 5. Type Screen.



Selection #2

Restaurant Price

Very High

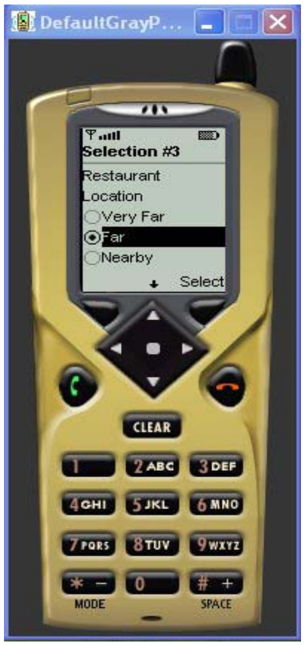
High

[✓ Medium]

Low

Select

Figure 6. Price Screen.



Selection #3

Restaurant Location

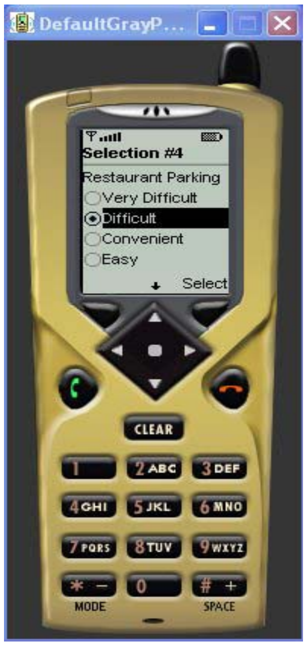
Very Far

[✓ Far]

Nearby

Select

Figure 7. Location Screen.



Selection #4

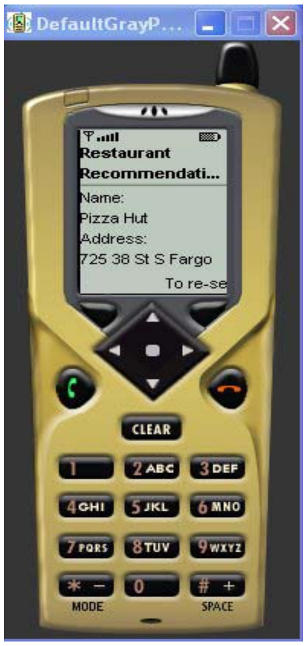
Restaurant Parking

Very Difficult [✓ Difficult] Convenient Easy

Select

Figure 8. Parking Screen.

In this case, the user likes a restaurant with Italian food and medium price, but does not care about the location and parking. By comparing the available restaurants, Pizza Hut is selected instead of Olive Garden. The main reason is the price. The result is shown in Figure 9. If the user changes his/her idea, he/she can simply click the Select button to continue the same process again. This time another restaurant is recommended as in Figure 10, i.e., Applebee. In the meantime, the user's selection will be written to the Profile table which reflects the user's most recent preference.

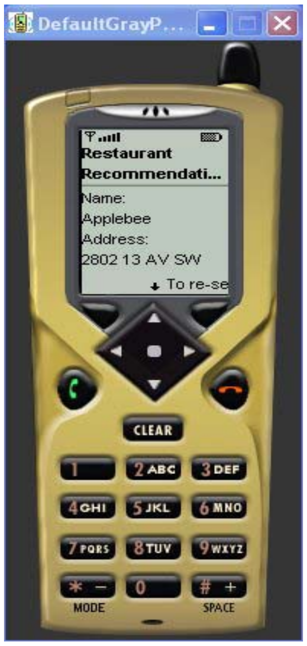


Restaurant Recommendati...

Name:
Pizza Hut
Address:
725 38 St S Fargo

To re-se

Figure 9. Recommendation – Pizza Hut.



Restaurant Recommendati...

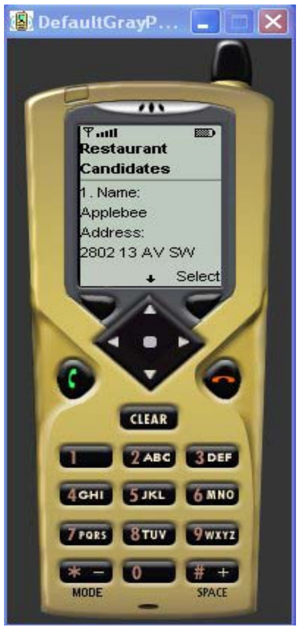
Name:
Applebee
Address:
2802 13 AV SW

To re-se

Figure 10. Recommendation – Applebee

4.3 Example for a Previous User

Another example is when the user starts the system with previous records already in the system. At the Candidate screen, the system will give the user a candidate list of at most three restaurants. These restaurants are those the user selected lately or the most frequently selected. Now we continue the scenario from the example in Section 4.2. The Applebee restaurant should be provided as the first candidate in the candidate screen. This is proved correct in Figure 11. The candidate list will be updated each time the user makes choices. In Figure 12, Olive Garden has been inserted as the first one in the restaurant candidate list.



Restaurant Candidates

1 . Name:
Applebee
Address:
2802 13 AV SW

Select

Figure 11. Candidate – Applebee.

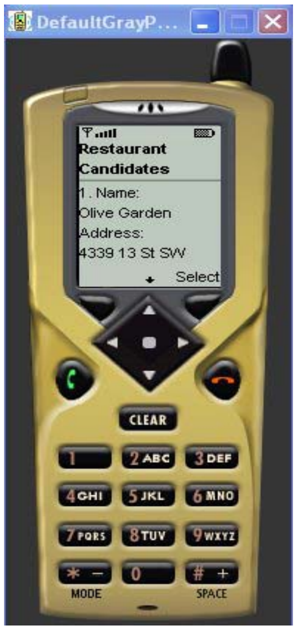


Figure 12. Candidate – Olive Garden.

5. CONCLUSIONS AND FUTURE WORK

Agent-based computing has been emerging as a major programming paradigm for the future. The Belief-Desire-Intention (BDI) agent modeling provides an efficient technique to build complex and intelligent system with its ability to learn and adapt and its characteristic for modular design. However, there are not many practical applications based on the BDI-agent concept developed yet. In this project we present an experimental framework for a Restaurant Finder system using BDI model and Java technology. We demonstrate how to design and implement the agent-based system with the help of database system. The system is able to learn from previous experiences and adapt to the changing environment.

In summary, the merit of this work is to show how to design a real-world application based on the BDI-agent model. This work shows how to implement very practically the BDI-agent-based application using the limitation of the current technology available to represent the BDI agent-based model appropriately. The noble idea of a belief-intention mapping table allows us to make the system reflective and intelligent by learning and adaptation. This work shows how to realize the theoretical BDI agent model, and how this realization can be used to relay information to a real-world application implementation.

There is a great risk that run-time dynamic mapping will affect the performance of the system. This is still an important issue for the BDI agent-based modeling. Java does not support directly run-time knowledge management or function implementation. It is a big challenge to handle how to update or add the Intention plans at run-time without affecting the system. If this can be done, the advantages of BDI agent-based modeling will be fully realized. This is part of the work for agent modeling to be completed in the future.

6. REFERENCES

[1] Agent Oriented Software Pty. Ltd., *JACK Intelligent Agents User Guide*, URL = www.agent-software.com.au, 1999.

[2] Bratman, Michael E., *Intention, Plans, and Practical Reason*, Harvard University Press, Cambridge, MA, 1987.

[3] Busetta, Paolo and Ramamohanarao, Kotagiri, *an Architecture for Mobil BDI Agent*, Mobile Computing Track, ACM SAC' 98, 1998.

[4] Deitel, Paul J., Harvey M., Deitel, Santry, Sean E., *Advanced JavaTM 2 Platform – How to Program*, Prentice Hall, Upper Saddle River, NJ, 463, 531-533, 543-549, 718-739, 755-784, 2002.

[5] Einhorn M. Jeffery and Jo, Chang-Hyun, A BDI Agent Software Development Process, University of North Dakota, 2002.

[6] Feng, Xin and Jo, Chang-Hyun, *Agent-based Stock Trader*, Department of Computer Science, University of North Dakota, 2002.

[7] Georgeff, Michael, Pell, Barney, Pollack, Martha, Tambe, Milind, and Wooldridge, Michael, *The Belief-Desire-Intention Model of Agenc*y (1999), Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98), URL = citeseer.nj.nec.com/georgeff99beliefdesireintention.html.

[8] Jo, Chang-Hyun, *A Seamless Approach to the Agent Development*, ACM SAC 2001, Las Vegas, NV, 641-647, 2001.

[9] Jo, Chang-Hyun and Arnold, Allen J., *the Agent-based Programming Language: APL*, ACM SAC 2002, Madrid, Spain, 27-31, 2002.

[10] Rao, Anand S. and Georgeff, Michael P., *BDI Agents: From Theory to Practice*, Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June 1995.