Extending the Sloan E-Commerce Project

with Intelligent User Interactions

by

Xingheng Wang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 3, 1999

© Copyright 1999 Xingheng Wang. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

February 3, 1999

Certified by

Glen L. Urban
Professor of Management
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

WITHDRAWN

JUL 15 1999

ENGLIBRARIES
Extending the Sloan E-Commerce Project
with Intelligent Customer Interactions

by

Xingheng Wang

Submitted to the
Department of Electrical Engineering and Computer Science

February 3, 1999

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Extensions to Trucktown, an e-commerce site, are designed and implemented. The
previous version of Trucktown is described. Issues such as distributed client/server
architecture, knowledge representation, customer behaviors and artificially intelligent
truck advisor are discussed. Based on a Bayesian Net, the truck advisor recommends
trucks based on supporting models and the customer’s responses to a series of questions.
The new extensions to the truck advisor enable more engaging interactions between the
customer and the truck advisor. A more flexible graphical customer interface is
prototyped. New algorithms and heuristics for determining the most important questions
to ask the customer are developed.

Thesis Supervisor: Glen L. Urban
Title: Professor of Management, MIT Sloan School of Management
Acknowledgements

This thesis is dedicated to my parents, Professor Shiquan Wang and Mrs. Jiazhen Dan of Utah State University, Logan, Utah. They have been supportive of my academic efforts throughout my life. Without their love and encouragement, this thesis and my forthcoming graduation would not have been possible.

I want to thank my best friend, Beau Smith, for all of the help, guidance and knowledge he has passed on to me during our four year friendship. He has helped me through some of the most difficult times in my life and been a constant source of encouragement and friendship. I feel privileged to know him, and look forward to further strengthening our friendship as we venture out into the real world.

I am deeply grateful to Professor Glen Urban, for his guidance and for having given me the opportunity to work on this interesting project. Despite his very busy schedule, he made himself available to advise me and work very closely with me to make progress on the research.

I thank other members of the E-Commerce research team at the MIT Sloan School: Ken Lynch, Frank Days, Andy Tian, Irene Wilson, and Brian Bower. The project has been a great example of team effort. It has been a pleasure to work with everyone. Special gratitude goes to Ken, with whom I worked very closely on this project until his graduation. He laid the foundations of this research project that were crucial to my thesis.

I also want to thank Professor Dennis Freeman for his academic guidance, Professor John D. Little for attending my oral presentation of this thesis, and Bryant Vernon for helping me edit this thesis.
1. Introduction

1.1 Background

E-commerce emerged as an industry only a few years ago, yet it has already experienced tremendous growth. As the number of Internet customers has increased exponentially, so have the numbers of electronic commerce sites and the numbers of purchase transactions made on the web. Not surprisingly, many traditional retailers have suddenly realized that they need an 'Internet Strategy,' which for many firms consists merely of digitizing their catalog and registering a domain name. Yet it is clear that the naive application of traditional static retailing methods to this new medium cannot sustain competition; with the competition just one click away, competition within the e-commerce industry is fiercer, and the customer is more demanding. In contrast to traditional stores, where there are costs to switch to a different store, i.e. transportation and time, if the customer is dissatisfied with any part of the on-line shopping experience, the customer can switch to a different on-line store at virtually no extra cost. There are several factors that customers use to decide where to buy: price, quality, number of selections, trustworthiness, and expert advice (Urban, 1998). Some leading sites have identified new factors that affect the customer's shopping experience, which include added features such as expert advice listed on a Web page, larger selection, or agents who browse the Web searching for the lowest price (Urban, 1998). Even with all these features, on-line shopping sites often lack smart interactions between an advisor and the customer.
1.2 Purpose

An expert advisor that recommends trucks for customers was implemented and deployed within the Trucktown version 2.0 in May of 1998. An extensive usability survey was conducted, and the expert advisor was well received. The purpose of this project is to extend the expert advisor and the e-commerce site to have more engaging interactions with the customer.

1.3 Organization

This thesis is divided into three major parts. Section 2 of this thesis gives an overview of the previous version of the e-commerce site, with emphasis on the architecture and the knowledge representation issues of the expert advisor. In section 3 of this thesis, three major extensions, Unfeasibility Expert, Dynamic Questions Expert, and Adaptive Graphical Customer Interfaces, are described. Section 4 concludes this thesis with a recount of the lessons learned and a discussion of future research.
2. Previous Work

2.1 Customer Model and Assumptions

For the expert truck recommendation engine, there are a lot of assumptions about customer behavior that need to be clarified. This section will present a model of customer behavior in the context of the Sloan Internet Truck Shopping Site.

2.1.1 Customer Purchase Model

The process in which the customer makes a decision to purchase a truck is modeled in six stages. 1) Need Arousal, 2) Channel Selection, 3) Information Search, 4) Consideration Set, 5) Purchase Decision, 6) Fulfillment (Benabadji, 1997).

Figure 1: Customer Purchase Model
The expert advisor system is mainly concerned with helping a customer decide which truck to buy. The expert advisor focuses on the transition from the information gathering stage to the consideration set stage. We make the following assumptions:

- Customers have some idea of what their needs are. It is reasonable to ask about those needs, because the customers already have need arousal before they choose our sales channel.
- Customers may not have adequate information about trucks and their specifications. This assumption is valid because we are assuming that they are in the stage of gathering information.
- Customers are logical and reasonable, but not perfect. This means that the customer is trying to be logical and reasonable; but sometimes, due to lack of information or otherwise, they may be persuaded by features or other factors. For example, the customer may only want to drive the truck in town. In that case, he may not need a four-wheel drive truck, but he may request one because the feature sounds more appealing.

2.1.2 Assumptions about Customer Trust

There are several different types of trust: channel-based trust, which is the degree of trust the customer has for the channel, outlet, or store of purchase; agent-based trust, which is the trustworthiness of the salesman or advisor when gathering information; information-based trust, which is the amount of trust the customer has for the information gathered;
fulfillment-based trust, which is whether the product fulfills the customer’s expectations (Benabadji, 1997).

On Trucktown, the expert advisor is the agent, and our system is expected to provide information to the customer about the trucks. Regular car dealers gain the customer’s trust by the appeal of personal characteristics such as clothing, general attitude, and tone of voice. The expert advisor in Trucktown, we assume, will gain the customer’s trust by the appeal of the characteristics such as clarity of explanations and reasons behind recommendations, revelation of inadequacies of certain recommendations, engaging and varied conversation, and providing customer feedback (Benabadji 1997).

2.2 System Architecture

The Trucktown version 2.0 was completed in May 1998 by a team of two MBA students and four computer science students, myself included. Like the previous version, the software is implemented in Java. Andy Tian developed the majority of the graphical customer interfaces, most of the virtual show room, and the underlying environment of the system. Ken Lynch designed and implemented the majority of the advisor recommendation engine for trucks. I designed and implemented the three-tier distributed client/server architecture, knowledge database, knowledge representation, and portions of the advisor recommendation engine.
Trucktown has a distributed three-tiered architecture. The technology used to set up the client/server architecture is Java RMI, that is, Remote Method Invocation. RMI is a protocol that facilitates communication between the server and the client. Java objects that implement the remote interface can reside on a server. When the client has a handle on the remote object (obtained through the RMI Naming Service provided by the Java RMI Registry), the client can call the methods of the remote objects that reside on another machine. The server uses JDBC:ODBC, which stands for Java Database Connectivity and Object Database Connectivity, to connect to the relational database. The JDBC is a wrapper around the ODBC for Java objects.

2.2.1 Client

The first tier of the three tiers is the client, which is the part the customer sees. On Trucktown, the client is the applet that runs in a Netscape browser. Unlike most other web sites, we follow a thick client model to take advantage of the increased computing power of personal computers. In a thick client model, the client does most of the computation, as opposed to a thin client model, in which the server does most of the computations. The client applet embodies the majority of the expert advisor and the entire front end customer interface. The disadvantage of a thick client is that it takes longer to download than a thin client does. However, the thick client model lightens the load on the server, so the server will be able to service more clients.
2.2.2 Server

The second tier is the server. The server is designed to handle a relatively large number of requests. It runs as a daemon (a program that constantly waits for client requests and handles them) on a Windows NT Network. It retrieves information regarding the trucks from the relational database using SQL statements. It constructs objects that make retrieving data easier for the client. Upon request from a client, the server will serialize the objects that contain the data and send them to the client.

2.2.3 Database

The third tier is a relational database. We chose to use the Microsoft Access database, because we were creating a prototype, and we didn’t want to spend the extra money on a more expensive database. The database is organized into several separate tables, each of which corresponds to one portion of the knowledge base. The ODBC driver is installed on the Windows NT Server computer. Section 2.3 will discuss in detail the design of the database in the context of knowledge representation for the expert system.

2.3 Knowledge Representation

The goal of knowledge representation is to put the knowledge into a format that can be easily used by the expert advisor to generate recommendations for trucks. The design
goals are simplicity, extensibility, and transferability. The database will be modified and updated by a human expert in truck sales and markets; therefore, the knowledge representation must be simple enough that someone who is unfamiliar with computer languages can still read and update it; the database must be easily extended to add or change the knowledge of the system; and the knowledge representation format should be easily transferred to different sets of knowledge.

The majority of the knowledge is represented by tables in a relational database. The table format accomplishes all three design goals. In a rule-based form of knowledge representation, the list of rules can get long, and the interactions between the rules (some rules may affect others) can get very complicated such that only professional computer programmers can understand the rules. In contrast to the rule-based format, the table format is simple and easy to read, even for those with little computing experience. Because of a table’s readability, and because most people are familiar with its format, it is much easier to add new entries to a table than to a rule-based system. Since the database stores much of the knowledge base that is going to be used in the Truck Recommendation Engine, the database is designed to represent knowledge in ways that can be easily accessed by the engine using SQL queries. The table format can represent knowledge about trucks or about any other product.

There are four layers of knowledge that relates to the questions that the expert advisor asks the customer. Each layer will be discussed in more detail in each of following sections.
2.3.1 Knowledge of Correlation of Responses

This knowledge base knows that the responses to certain questions are correlated with each other. The questions are grouped into correlation groups. If one question is in a group all by itself, then it is independent of, or has very little correlation to, all the other questions. If two or more questions are correlated with each other, then they are in the same correlation group, and they are correlated with each other. Since the knowledge set is small, it is stored in the form of rules. This knowledge is hardcoded into a Java class. In the example table below, if two responses are in the same correlation group, they are considered correlated with each other.

<table>
<thead>
<tr>
<th>Responses</th>
<th>Correlation Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full/compact</td>
<td>Full/Compact</td>
</tr>
<tr>
<td>Number of people to carry</td>
<td>Number of people to carry</td>
</tr>
<tr>
<td>Drive on rough road</td>
<td>Off road driving</td>
</tr>
<tr>
<td>Off road driving</td>
<td>Off road driving</td>
</tr>
<tr>
<td>Towing a trailer</td>
<td>Towing</td>
</tr>
<tr>
<td>Commercial hauling</td>
<td>Hauling</td>
</tr>
<tr>
<td>Home supply hauling</td>
<td>Hauling</td>
</tr>
<tr>
<td>Driving on snowy or icy road</td>
<td>Off road driving</td>
</tr>
<tr>
<td>Fishing or Hunting</td>
<td>Off road driving</td>
</tr>
<tr>
<td>Construction</td>
<td>Construction</td>
</tr>
</tbody>
</table>

In Table 1, for example, if the customer answers that he likes fishing or hunting and he likes driving on rough roads, then these two answers are not independent. They both imply that the customer mostly likely will use the truck for off-road driving.
2.3.2 Knowledge of Truck Utility Rankings

The Truck Utility Ranking information is obtained through past marketing research. Each truck is ranked on five dimensions: value, fuel economy, dependability, power, and safety. This information is stored as a table in the Microsoft Access database. The MSRP is the ranking of the price against the value of the truck. The fuel economy is based on the fuel efficiency of the truck in comparison to other trucks. The dependability is based on the likelihood of the truck to break down. The safety ranking is based on the number of safety features on the truck. All the rankings are standardized.

Table 2: Partial View of the Truck Utility Ranking Table

<table>
<thead>
<tr>
<th>Vehicle</th>
<th>MSRP_Correct</th>
<th>Fuel_Economy_Correct</th>
<th>Dependability_Correct</th>
<th>Horsepower_Correct</th>
<th>Safety_Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford_F350_(2WD)</td>
<td>-0.03</td>
<td>-0.76</td>
<td>0.4</td>
<td>0.35</td>
<td>1.42</td>
</tr>
<tr>
<td>Ford_F350_(4WD)</td>
<td>0.8</td>
<td>-1.32</td>
<td>0.4</td>
<td>0.35</td>
<td>1.2</td>
</tr>
<tr>
<td>Ford_F350_Crew_Cab_(2WD)</td>
<td>0.87</td>
<td>-0.76</td>
<td>0.4</td>
<td>0.35</td>
<td>1.17</td>
</tr>
<tr>
<td>Ford_F350_Crew_Cab_(4WD)</td>
<td>1.7</td>
<td>-1.32</td>
<td>0.4</td>
<td>0.35</td>
<td>0.82</td>
</tr>
<tr>
<td>Ford_F350_Extended_(2WD)</td>
<td>0.81</td>
<td>-1.04</td>
<td>0.4</td>
<td>0.35</td>
<td>1.69</td>
</tr>
<tr>
<td>Ford_Ranger_(2WD)</td>
<td>-1.94</td>
<td>1.78</td>
<td>-1.64</td>
<td>-1.19</td>
<td>-0.65</td>
</tr>
<tr>
<td>Ford_Ranger_(4WD)</td>
<td>-0.47</td>
<td>0.65</td>
<td>-1.64</td>
<td>-1.19</td>
<td>-0.72</td>
</tr>
<tr>
<td>Ford_Ranger_Extended_(2WD)</td>
<td>-1.03</td>
<td>0.94</td>
<td>-1.64</td>
<td>-1.19</td>
<td>-0.72</td>
</tr>
<tr>
<td>Ford_Ranger_Extended_(4WD)</td>
<td>-0.03</td>
<td>0.65</td>
<td>-1.64</td>
<td>-1.19</td>
<td>-0.8</td>
</tr>
<tr>
<td>GMC_Sierra_C1500_(2WD)</td>
<td>-0.95</td>
<td>0.09</td>
<td>0.35</td>
<td>0.54</td>
<td>-0.19</td>
</tr>
<tr>
<td>GMC_Sierra_C1500_Extended_(2WD)</td>
<td>-0</td>
<td>0.37</td>
<td>0.35</td>
<td>0.54</td>
<td>-0.34</td>
</tr>
<tr>
<td>GMC_Sierra_C2500_(2WD)</td>
<td>-0.17</td>
<td>-0.19</td>
<td>0.35</td>
<td>0.54</td>
<td>0.33</td>
</tr>
<tr>
<td>GMC_Sierra_C2500_Extended_(2WD)</td>
<td>0.5</td>
<td>-0.19</td>
<td>0.35</td>
<td>0.54</td>
<td>0.86</td>
</tr>
<tr>
<td>GMC_Sierra_C3500_(2WD)</td>
<td>0.1</td>
<td>-0.76</td>
<td>0.35</td>
<td>1.17</td>
<td>1.28</td>
</tr>
<tr>
<td>GMC_Sierra_C3500_Crew_Cab_(2WD)</td>
<td>0.95</td>
<td>-0.76</td>
<td>0.35</td>
<td>1.17</td>
<td>0.74</td>
</tr>
</tbody>
</table>
2.3.3 Knowledge of Bayesian Probabilities

The knowledge of the conditional probabilities of responses given the truck purchased is the main knowledge base of the system. Each entry in the table is $P(R_{r,q}|A_a)$, i.e., the probability that the person will answer response $r$ to question $q$ given that their true preference is the truck alternative $a$.

The conditional probabilities are determined from past marketing data from secondary sources and the human expert's understanding of the truck specifications and truck markets. The expert in our case is Frank Days, who has an MBA in marketing. The human expert is given an empty table and he enters the data that captures all his knowledge about the trucks.

Table 3: Partial View of the Bayesian Probability Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevy C1500 (2WD)</td>
<td>20</td>
<td>60</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Chevy C1500_Extended (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy C2500 (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>9</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Chevy C2500_Extended (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy C3500 (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy C3500_Crew (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy C3500_Extended (2WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K1500 (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K1500_Extended (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K2500 (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K2500_Extended (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K3500 (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K3500_Crew (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy K3500_Extended (4WD)</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Chevy_5-10 (2WD)</td>
<td>60</td>
<td>20</td>
<td>60</td>
<td>20</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Chevy_5-10 (4WD)</td>
<td>60</td>
<td>20</td>
<td>1</td>
<td>9</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Chevy_5-10_Extended (2WD)</td>
<td>60</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Chevy_6-10 (2WD)</td>
<td>60</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Dodge_Dakota (2WD)</td>
<td>60</td>
<td>20</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>10</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

The table above is read in the following way: each numeric entry is a conditional probability in terms of percentages. For example, there is a number 20 at the entry where the horizontal row is Chevy C 1500 (2WD), and the vertical column is compact. The number 20 means that if the customer's true preference is the Chevy C 1500 (2WD), there
is a 20 percent probability he will say that he wanted a compact size truck when asked his preference for size.

2.3.4 Knowledge of Consultative Messages

The system must know how to explain in a consultative manner why it recommends a truck. The database stores knowledge about what to say when delivering reasons for why it recommended the truck it did. The messages are represented as strings stored in a table format and are associated with each of the customer responses. They can be used to construct sentences for conversation with the customer later on.

<table>
<thead>
<tr>
<th>key</th>
<th>compact</th>
<th>full</th>
<th>Price_10_12k</th>
<th>Off_road_no</th>
<th>Off_road_yes</th>
<th>Towing_no</th>
<th>Towing_yes</th>
<th>Hauling_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>What to say when this truck is recommended for this reason.</td>
<td>This is a compact truck and you stated that you prefer full size trucks</td>
<td>This truck is within your preferred price range</td>
<td>You said that you prefer trucks that are not designed for offroad driving</td>
<td>You said that you prefer trucks that are designed for offroad driving</td>
<td>You said that you prefer trucks that are not designed for towing</td>
<td>You said that you prefer trucks that are designed for towing</td>
<td>You said that you prefer trucks that are not designed for hauling things</td>
<td></td>
</tr>
<tr>
<td>What to say when there is a problem with the recommended truck.</td>
<td>This is a full size truck and you stated that you prefer full size trucks</td>
<td>This truck is not within your preferred price range</td>
<td>This truck is not designed for offroad driving</td>
<td>This truck is not designed for offroad driving</td>
<td>This truck is not designed for towing</td>
<td>This truck is not designed for towing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each possible response to the questions is associated with two strings. The sales features are what the expert say when the truck it recommends satisfies the customer’s preferences, and the sales problems are what the expert says if the truck it recommends
does not satisfy the customer's preferences. These strings are used to construct meaningful feedback to the customers.

2.4 AI Configuration Engine

In his thesis, Ken Lynch detailed implementation and design description of the Needs-Based AI Configuration Engine. The information received from the customer and from the database is passed to the expert system through several layers: Response De-Correlator, Utility Analysis, Bayesian Expert, and Consultative Sales Expert (Lynch 1998). I implemented the Response De-Correlator and portions of the Consultative Sales Expert.

2.4.1 De-Correlator

The Response De-Correlator consolidates correlated responses into one response that has minimal correlation with other responses. The consolidated response can be used as inputs to the Bayesian Expert, which assumes that all inputs are statistically independent from each other (Lynch, 1998). The De-Correlator is a rule-based system. The rules are hard coded in procedures in a series of “if” statements. For example, if the customer says he wants to use the truck to drive on rough roads, and then says he wants to use the truck for hunting or fishing, it is highly likely that he wants to use the truck for the same purposes. It makes sense, then, to consolidate the two responses into one response, i.e.
that the customer wants to use the truck for off-road driving. Similarly, all the responses that belong to the same correlation group will be consolidated into one response.

2.4.2 Utility Analysis

The Utility Analysis is based on knowledge of the truck utility rankings from previous marketing research. The truck’s a-priori probabilities for being a good recommendation choice will be set using the result of the Utility Analysis.

From the utility rankings, we can get a-priori probabilities using the following formula (Urban 1998):

\[
P(A_a) = \frac{\exp \sum_j \alpha_d D_{d,a}}{\sum_a \exp \sum_j \alpha_d D_{d,a}}
\]

\(P(A_a)\) = Probability of purchase of product alternative \(a\), i.e. the a-priori probability

\(\alpha_d\) = constant sum importance of dimension \(d\) given by customer

\(D_{d,a}\) = standardized database value of dimension \(d\) for alternative \(a\) (Urban 1998), i.e. the values stored in the utility knowledge database

It is debatable that using the utility ranking is the correct way to approximate a-priori probabilities for purchasing trucks. In the customer purchase model, however, we assumed the customer is rational. By entering the utility preference trade-off values into
the five different dimensions, he would prefer trucks with a ranking that most closely matches his ratings. A rational person would most likely purchase such a truck alternative. Therefore, we believe the utility analysis is a reasonable way to approximate the probabilities.

2.4.3 Bayesian Analysis

The Bayesian Analysis will be the main tool used in generating a consideration set of trucks for the customer. The Bayesian Analysis Expert tries to find the set of four trucks that best match the customer preferences in terms of the features and the properties of the trucks. The expert will ask the customer a series of questions regarding the usage of the trucks and customer preferences. In our customer purchase model, we assumed that the customer could be persuaded by features and requests. (For example, a customer may not need a 4-wheel-drive truck, but he might still prefer it). Although a perfectly rational customer would chose the truck that most closely suits his utility needs. The customer often has specific requests of the trucks that might influence his decision. In our case, we believe a Bayesian net is the model that best captures this influence.

A Bayesian net underlies the recommendation engine. To simplify the net, we assume that the inputs into the network are not correlated, which is a valid assumption because the De-Correlation Expert aggregates the highly correlated responses into a single response that has a minimum correlation with other responses.
The following are the steps taken to update the probability of all trucks for each new response input (Lynch 1998).

Step 1: We retrieve the $P(R_{r,q} | A_a)$, i.e. the conditional probability of answering question $q$ with response $r$ given that the purchased alternative is $a$. These data are stored in the knowledge database of Bayesian Probabilities.

Step 2: We first compute the marginal probabilities.

$$P(R_{r,q}) = \sum_a P(A_a) P(R_{r,q} | A_a)$$

$P(R_{r,q})$ = the marginal probability that a customer answers question $q$ with the response $r$ (Urban 1998).

Step 3: Use the Bayes Theorem to create the new purchase probabilities (Urban 1998):

$$P'(A_a) = P(A_a | R_{r,q}) = \frac{P(A_a) P(R_{r,q} | A_a)}{P(R_{r,q})}$$

$P(A_a)$ = the prior probability of purchase of truck alternative $a$ before the question $q$ is answered.

$P(A_a | R_{r,q})$ = the conditional probability that the customer will purchase the truck alternative $a$ given that he answered with response $r$ to question $q$. 

19
However, since the question q is already now answered, the $P'(A_a)$ becomes the new probability of purchasing the truck alternative $a$. The probability can be iteratively updated across all responses to questions.

With the updated probabilities, the list of trucks is sorted according to the probabilities. The top four trucks with highest probability are chosen as the recommended consideration set (Lynch 1998). The reason is that, as with our customer model, we don’t want to make the decision for the customer; we want to help the customer find the consideration set so he can go to the show room and make a final decision.

2.4.4 Consultative Sales Expert

The Consultative Sales Expert constructs reasons why the customer should buy the trucks it has recommended (Lynch 1998). As presented in the trust model, explanation of the recommendations is an important trust building factor for our agent. The Consultative Sales Expert assumes the recommendations are made already, and determines which reasons are suitable for the current trucks recommended. It uses the knowledge of the consultative messages stored in the database to create the recommendation reasons. The expert engine figures out which question changes the probability the most, given the truck recommended. The highest ones are chosen as the most important reasons.

For each answer question pair, $r$ and $q$, an influence factor delta, $D(R_{r,q})$ is calculated for recommendation of a truck by this formula:
\[ D(R_{r,q}) = P(A_d|R_{r,q}) - P(A_d) \]

Where \( P(A_d) \) is the a-priori probability before any of the questions are answered, i.e. it is the original a-priori probability computed from Utility Analysis.

The top three responses with highest influences are chosen as reasons why the truck is recommended, and the response with the highest negative influence is selected as the biggest sales problem.

Once the sales features and sales problem are found, then the consultative selling expert will be able to show the customer the reasons by retrieving the strings in the Consultative Message Database associated with the responses (Lynch 1998).

3. Extensions

Extensions to the e-commerce site mostly involve making the Expert Advisor smarter and more engaging with the customer. I implemented the majority of the extensions. Brian Bower made some contributions.

3.1 Unfeasibility Expert

3.1.1 Motivation and Background
The Unfeasibility Expert will try to alert the customer when something the customer requested is inherently impossible to achieve. From the customer purchase model, we indicated that we are going to assume that the customer is logical, but does not have all the information. It is conceivable that the customer may not know that some pairs of requests are impossible to meet. Once the advisor explains the unfeasibility of certain requests, the customer might reconsider and make a more informed decision. In the customer trust model, one of the trust cues for the agent is engaging and informative feedback. When the expert explains the conflicts in the customer’s responses, it will be engaging in more intelligent interaction, which makes the customer aware that the expert is taking his concerns into consideration. That will encourage the customer to trust the expert’s recommendations.

3.1.2 Design

For simplicity, the Unfeasibility Expert reuses much of the code of the Bayesian Analysis Expert to make the calculations. For ease of integration, the Unfeasibility Expert will not keep memories of past calculations. For portability, a general algorithm is designed.

3.1.3 Algorithm

The Unfeasibility Algorithm is based on the Bayesian Analysis. Given a pair of the questions answered, if both responses cause the probability of a truck to change in the
same direction, i.e. to both increase or to both decrease, this implies that there exists some truck that satisfies the combination of request from the customer. The two responses are considered consistent with each other. If the two responses are inconsistent with each other for all truck alternatives, then the combination of requests is impossible to achieve with any truck alternative. The pair of responses will be classified as an unfeasibility. To be more specific, the algorithm follows these steps:

Step 1: For each truck, the expert first determines out the baseline probability for buying alternative a, let that be \( P_b(A_a) \). The Baseline probability is the same as the a-priori probability obtained from Utility Analysis.

Step 2: For each response \( r \) to question \( q \), do the Bayesian update (similar to the step in the Bayesian Expert) to find out the conditional probability \( P(A_a | R_{r,q}) \) using the baseline probability determined in step 1.

\[
P(A_a | R_{r,q}) = \frac{P_b(A_a)P(R_{r,q} | A_a)}{P(R_{r,q})}
\]

Step 3: Calculate the influence of each of the response-question pairs for one particular truck alternative. Then for a responses \( R_{r,q} \), define the influence to be delta:

\[
D(A_a, R_{r,q}) = P(A_a | R_{r,q}) - P_b(A_a)
\]

where \( P_b(A_a) \) is the baseline probability. We calculate the deltas for each response \( r \) and each truck alternative \( a \).

Step 4: Fill in Consistency Matrixes.
For each truck alternative $A_a$, there will have a matrix called Consistency Matrix. The matrix is indexed by the customer-inputted responses. For every pair of responses $R_{r1,q1}$ and $R_{r2,q2}$, if $D(A_a|R_{r1,q1})$ and $D(A_a|R_{r2,q2})$ are both positive or both negative, i.e. that the responses $R_{r1,q1}$ and $R_{r2,q2}$ are consistent for the truck alternative $A_a$. If the deltas are of the same polarity, then that implies that the $R_{r1,q1}$ and $R_{r2,q2}$ influence the probability of this truck in the same direction, therefore the truck satisfies both responses. On the other hand, if $D(A_a|R_{r1,q1})$ and $D(A_a|R_{r2,q2})$ have different polarity, then that implies the responses $R_{r1,q1}$ and $R_{r2,q2}$ influence the probability of the truck in different directions, which implies this particular truck doesn’t satisfy both responses.

Step 6: Detect Unfeasibility.

If $D(A_a|R_{r1,q1})$ and $D(A_a|R_{r2,q2})$ have different polarities for all truck alternatives, that implies that the response $R_{r1,q1}$ and $R_{r2,q2}$ is a combination for which no truck can satisfy the requirements. Thus we have an unfeasibility.

Step 7: Measure the Degree of Unfeasibility.

In this step, the Unfeasibility Expert uses a heuristic to measure the size of the inconsistency. For each of the unfeasible pairs of responses $R_{r1,q1}$ and $R_{r2,q2}$, the inconsistency measure function is $I(R_{r1,q1}, R_{r2,q2})$.

$$I(R_{r1,q1}, R_{r2,q2}) = \sum_a (|D(A_a, R_{r1,q1})| + |D(A_a, R_{r2,q2})|)$$
This works because \( D(A_a, R_{r,q}) = P(A_a R_{r,q}) - P_b (A_a) \), and we know that for the inconsistent pair, the signs of the two deltas are opposites. Thus, the sum of the absolute value of the deltas would be the distance between the two probabilities. The Unfeasibility Expert assumes that a bigger the distance implies more unfeasibility.

Step 8: Sort and pick the highest unfeasible pair of responses to display to the customer, and ask if the customer want to reconsider.

3.1.4 Implementation Notes

Every time the customer answers a question, and before the question panel displays the next question, it will call the Unfeasibility Expert to decide for unfeasibilites. The unfeasibilities are encapsulated by the Java class ConsistencyInfo. The ConsistencyInfo data class contains four fields: the pair of question numbers that resulted in an unfeasibility, a Boolean value to indicate if the pair of questions are consistent with each other, and the \( \text{SumOfAbsoluteValue} \) field, which is a measure of the degree of unfeasibility.

The system also keeps an inconsistency history. The advisor gives the customer the option of ignoring the unfeasibility. If the customer chooses to ignore the unfeasibility, the fact is stored in the inconsistency history, so that it isn’t brought up again.
3.1.5 Results and Issues

The following screen shots give a real scenario of the Inconsistency Expert at work. In this example, the customer originally indicated that he wanted a compact size truck. Later on, the customer answered that he wanted to use the truck for hauling. Because compact size trucks are not designed for hauling, the expert asks the customer to reconsider. If customer clicks on Usage button, the Usage Question Panel will be displayed.

![Figure 2: Full or Compact Size Truck Question](image-url)
Figure 3: Usage of Truck Question

Figure 4: The Apology within the Unfeasibility Panel
The general algorithm for detecting unfeasibilities is sufficient for most cases. In many engineering problems, the more general the algorithm is, the less effective it is. There are some special cases of unfeasibility that may not make logical sense. For example, say the customer said that he won’t be using the truck for construction work, and he chooses a full size truck. The Unfeasibility Expert will raise an unfeasibility, because all full-size trucks are suitable for construction work. He is paying extra if he doesn’t use the full-size truck for construction work. On the other hand, if he prefers the full-size truck there is really no unfeasibility. These types of logical errors occur with most of the usage questions. Another type of logical error of unfeasibility is with brand names. For example, when a customer asks for a Dodge truck, and then asks for Toyota truck, the algorithm will detect an unfeasibility. There is no truck that is both manufactured by Dodge and Toyota. The general Unfeasibility Algorithm failed to realize that there is no contradiction, because the customer is asking for either a Dodge truck or a Toyota truck. An additional logic module was added to the Unfeasibility Expert to overcome the logical errors associated with these special cases. A procedure is implemented that removes unfeasibilities that do not make logical sense.
3.2 Dynamic Questions Expert

3.2.1 Motivation

In the previous version of the expert system, all the questions were asked in the same order regardless of how the customer responded to the questions, which is not be the most effective way of gathering information. In the new version, the Dynamic Questions Expert uses heuristics to determine which question to ask next, based upon the customer's answers to previous questions. Because the advisor is more engaging, the customer will likely feel that advisor is more personalized. The expert must also choose questions that lead to the best set of recommendations with as few questions as possible (Lynch, 1998).

3.2.2 Design

The dynamic questions are integrated with the Bayesian Expert, because it uses much of the same code that was originally used in the Bayesian Expert. The Dynamic Question Expert uses the same set of knowledge that the Bayesian Expert uses. Since the project is a prototype to test ideas and methodologies, one important design goal is to have a clear and simple implementation of the algorithm to test. The other design goals such as speed and efficiency are not as important. We designed the system to have no memory of the past results, so that the implementation can be simpler with the trade off that the system may need to do extra computations.
3.2.3 Dynamic Question Algorithms

The general Dynamic Questions Algorithm computes an importance factor for all the unanswered questions, and then chooses the question \( q \) with the highest importance factor. This section will present two of the heuristics used to find the importance factor.

3.2.3.1 Using Aggregate Influence of the Question as Importance Factor

The importance factor of a question is the aggregate degree of a question to cause the probability of purchasing a truck to shift. This particular follows these steps:

Step 1: Find the baseline probability \( P_b(A_a) \),

\[
P_b(A_a) = P(A_a | All the answered questions)
\]

Step 2: For each of the unanswered questions \( q \), try out response \( r \), and determine the conditional probability using the Bayes Theorem and the baseline probability. The influence of the particular response is

\[
|P(A_a | R_{r,q}) - P_b(A_a) |
\]

Step 3: Then the total influence \( I(q) \) of the questions \( q \) is the average of all the influence of all the possible responses summed over all possible truck alternatives.
\[ I(q) = \sum_{a}^{r} \frac{\sum_{r,q} |P(A_a | R_{r,q}) - P_b(A_a)|}{n} \]

\( n \) = the total number of possible responses.

Step 4: use the unanswered \( q \) with the highest \( I(q) \) as the next dynamic question.

3.2.3.1 Using Expect Utility of the Question as Importance Factor

The above algorithm is a straightforward way of finding the most important question. It uses the average amount of change in probability to determine which question will result in the greatest change in probability over all trucks. However, it may not be the best way, for two reasons. First, for a particular question, one response may be more likely to occur than others may. Second, one response may cause the highest change in probability, but it may not cause the highest change in probability for the truck that might give the most utility to the customer. The following algorithm addresses these two issues by using the expected value of the utility of the questions as a measure of importance.

Step 1: If the utility questions are answered already, we use the utility responses to derive the utility rankings of each truck (see section 5.3). This gives us: Utility, or \( U(A_a) \) of each truck. Otherwise, we assume the customer’s utility preferences in the five dimensions are equal; therefore, this algorithm is only effective after the utility preferences are entered.
Step 2: Use the Bayesian Analysis Expert to calculate the probability each truck has of being purchased given the vector of responses we have so far.

\[ P_b(A_a) = P(A_a | \text{All the answered questions}) \]

We are going to use this as the baseline probability \( P_b(A_a) \).

Step 3: To compute an expected utility \( EU(q) \) of question \( q \), we use a standard formula for computing expect values:

\[ EU(q) = \sum_r P(R_{r,q}) \times ERU(R_{r,q}) \]

\( ERU(R_{r,q}) \) = The expected response utility of the response \( r \) to the question \( q \).

\( P(R_{r,q}) \) = The probability of a customer answering response \( r \) to the question \( q \).

This probability is computed using the same formula as in Bayesian Analysis.

The expected utility of the response \( r \) to question \( q \) can be computed as follows.

\[ ERU(R_{r,q}) = \sum_a P(A_a | R_{r,q}) U(A_a) \]

\( P(A_a | R_{r,q}) \) = the conditional probability computed using the Bayes Theorem and the baseline probability \( P_b(A_a) \).

\( U(A_a) \) = the utility of the truck alternative \( a \), computed from the Utility Analysis.

\[ U(A_a) = \sum_d \alpha_d D_{d,a} \]

\( \alpha_d \) = the value of preference in utility dimension \( d \) given by the customer
\[ D_{d,a} = \text{standardized database value of dimension } d \text{ for alternative } a \]

To summarize the formulas: the Dynamic Questions Expert calculates the expected response utility of each possible response \( r \), which is the sum, over all trucks, of the utility of each truck weighted by the probability that each truck has of being purchased given response \( r \) to question \( q \). From the previous calculations, we obtain the expected utility of an unanswered question, which is the sum, over all responses, of the expected response utility weighted by the probability of that response \( r \) to question \( q \).

Step 4: The questions \( q \) with highest \( EU(q) \) is determined and selected from all the unanswered questions.

3.2.4 Issues with Dynamic Question Expert

In Trucktown, both Dynamic Questions Algorithms are implemented. Because we have to calculate the aforementioned probabilities for all trucks and for all possible unanswered questions for all possible responses, at each step of questionnaire, the number of calculations is very large. The speed of the Dynamic Questions Expert using the first algorithm is slow; the speed of the Dynamic Questions Expert using the second algorithm is extremely slow: the applet sometimes halts during some execution. For practical reasons, I decided to use the first algorithm for generating dynamic questions, despite the fact the second algorithm ultimately tries to maximize the utility of the recommended trucks.
3.3 Adaptive Graphical Customer Interface

3.3.1 Motivation

Different people have different preferences for interacting with the computer. Different preferences are the reason why different interaction instruments have been designed, such as the mouse, keyboard, and touch pad. Different people may have different preferences in interacting with a computer program or Internet site as well (Urban, 1998). For shopping, some people prefer to browse around, while others just want to go to the store, get what they want, and leave. In Trucktown, we want to have a flexible interface that accommodates both preferences.

Another reason that motivated us to develop the Adaptive Graphical Customer Interface on our Web site is that we want to give the perception that we have a large quantity of information. During customer usability testing, a lot of customers thought that Microsoft Carpoint had a larger quantity of information than the Trucktown (Urban 1998). The main difference between our site and the Microsoft site was that Microsoft’s Carpoint listed all the trucks at once, so the customer could look at all the truck specifications by clicking on the truck’s name. On our original site, if the customer wanted to look at the trucks, he had to go through our expert advisor before he could enter the showroom. Yet, some people knew exactly what model of truck they wanted, and they just wanted to get information about them. They didn’t want only to look at the trucks recommended by the advisor. By developing the Adaptive Graphical Customer Interface, we give them that option on Trucktown.
3.3.2 Design and Implementation

The main design goal was to design add-on interfaces that could integrate with the original Trucktown interface. The change from the old interface to the new interface should be gradual, such that the transition is smooth from one interface to another. The interface changes depending on how the customer interacts with the Web site.

A side frame, which contains a list of all trucks, is added to the side of the Trucktown map interface. As the customer progresses through the side bar, the original map of the Trucktown grows smaller and smaller, making room for detailed information about the selected truck. If the customer then decides to use the map to explore Trucktown, the original size of the map will be restored.

3.3.3 Results and Example

The original Trucktown navigation and interface is described in Andy Tian's thesis (Tian 1998). In this section, several screen shots step through a scenario that takes advantage of the Adaptive Graphical Customer Interfaces. Note that as the customer progresses through the side bar's list toward a final choice, the size of the original Trucktown map decreases.
Figure 5: Starting Interface with Original Trucktown Map Embedded

Figure 6: By Using the Sidebar, the Trucktown Map Shrink.
4. Conclusion

4.1 Separation of Knowledge and Mechanism

One major advantage of our expert system is our attempt to separate the knowledge from the algorithms that use that knowledge. The separation of knowledge from mechanism gives rise to a lot of flexibility. Generally, to implement an expert system, one needs a human expert in that field, knowledge engineers that gather the knowledge, and software engineers who write the code. Often, in systems whose structure resembles rule-based systems, the knowledge engineers have to know the software pretty well, and the software engineers have to understand much of the knowledge, because the knowledge and the code are so closely coupled together. The knowledge engineers have to
communicate the knowledge to the people. Our system makes knowledge engineering easier for the experts in the field. The expert on trucks and marketing, Frank Days, didn’t need to know anything about programming. The choice of using a Microsoft Access database makes knowledge entry very customer-friendly for the human expert. We just have to explain what each entry means in the database (probability of the response to a particular question, given the choice of a particular truck), and he can just enter the data by himself. Later on, if the expert changes his mind about any pieces of knowledge, he can just go back to the database and change the values without even consulting the knowledge engineers. On the other hand, the software engineers do not need to learn about a lot of trucks in order to implement the mechanism. For example, they don’t need to know the fact that if you want to drive a truck in the snow, you’d better have a four-wheel-drive truck. The numbers that the expert enters into the database captures that piece of knowledge. The project went more smoothly because the technical communication between the experts and software engineers was reduced.

An additional benefit of the separation of the knowledge from the mechanism is that the mechanism can be potentially ported with minimum or no modification to another knowledge base. For example, a database that contains knowledge on laptops, with little or no change to the algorithms, then the advisor can be used to recommend laptops.

On the other hand, there is no complete separation of the knowledge from the implementation of the algorithms. Often, when an algorithm is too general, it is less effective. With complex information, as demonstrated in the implementation of the
algorithm for detecting inconsistency, using the unfeasibilities generated by the recommendation engine does not always make sense logically. Therefore, a human with the logic will have to write code to circumvent the special cases.

4.2 Limitations and Future Research

4.2.1 Difficulty in Determining Independence of Responses

Because we used a Bayesian net to make recommendations, we had to use the noisy-or model to reduce the number of calculations. In the noisy-or model, the inputs to the Bayesian nets have to be nearly uncorrelated with each other. However, even with the De-Correlation Expert, it is very difficult to determine whether the two responses are correlated with each other or not. For example, we assumed that the height of the customer and the preferred price range are independent of each other. However, we can’t be sure there is absolutely no correlation between someone who is tall and someone who has more money to spend. In reality, there is actually a correlation. On average, taller people make more money than shorter people do, because society is biased against short people. However, there isn’t a large correlation, so we decided against making it part of our model, and we decided to ignore similar correlations in many other cases. It will be very difficult to measure the amount of error introduced into the computed probabilities by ignoring such small correlations.
4.2.2 More on Unfeasibility Expert

Currently, the Inconsistency Expert only detects inconsistencies between any two pairs of responses, but it does not detect inconsistencies that might be due to a combination of three or more requests. As the number of responses we consider increases, the number of combinations increases exponentially. The algorithm described for the pair case may not be extensible to combinations of three or more responses. Also, another challenge is that, as the number of multi-responses increases, it may be harder and harder to find trucks that satisfy all the requirements. Thus, the number of multi-response unfeasibilities may increase greatly and the customer may be frustrated by an expert that keeps asking him to reconsider his responses.

The Unfeasibility Expert can also serve as a new product research tool. The system has the capability of determining customer wants and compiling them into a list. Truck manufacturers can use such compilations to design better and more popular products. For example, GM missed a big market opportunity when it failed to address customers’ needs for a large sports utility luxury truck. A few years ago there were trucks like Toyota Tacoma that were small luxury trucks; there were small and non-luxurious Dodge trucks (Lynch 1998); and there were large trucks from GMC; however, there were no large luxury trucks in the market. Had GM the knowledge that a lot of people wanted trucks that were large and luxurious, a preference that would have been detected by the Unfeasibility Expert, it would not have missed this market (Lynch 1998).
4.3 Summary

This thesis outlines the client/server architecture and some of the features of an e-commerce site that sells trucks over the Internet. Using a model of customer behavior with explicit assumptions, the thesis describes an Artificially Intelligent Advisor that recommends trucks to the customers and gains trust from customers. The advisor is a Knowledge Based Expert System built on the Bayesian nets. Extensions to the advisor are designed and implemented to improve the interaction between the advisor and the customer. The new interface detects those customers who know exactly what model of truck they want, and provides a quicker way for them to get the information they need. Using a heuristic algorithm, the new expert will inform the customer if there is an unfeasibility in their answers to the questions, and will give them a chance their mind. The new expert will always try to ask the most relevant question depending on how the customer answered previous questions. Despite the fact that this particular advisor prototype recommends trucks, all the algorithms are designed such that they can be ported to provide recommendations for a wide array of products. In the near future, commercial versions of the virtual advisor based on our prototype could potentially provide an improved on-line shopping experience for customers.
Bibliography


Appendix

Below is the sample Java code that implements the algorithm that determines the most important question base on the expected utility of the unanswered questions.

```java
public int getNextUtilityDynamicQuestion(ResponsesDB rDB, Truck[] trucks) {
    int i, j;

    double[] expected_question_utility = new double[rDB.numberOfQuestions+1];
    expected_question_utility[0] = Double.NEGATIVE_INFINITY; //just a really small number

    for(i = 1; i <= rDB.numberOfQuestions; i++) {
        if (rDB.responseArray[i].response != -1) {
            //If a question's response is -1, that means it
            //is not answered yet. In this case,
            //the question is already answered;
            //so the utility of the question is negative infinity;
            expected_question_utility[i] = Double.NEGATIVE_INFINITY;
        } else {
            //the question is not answered
            //we need to compute the utility of the question at hand

            expected_question_utility[i] = 0;
            //sum up all the probabilities;
            int max_response = rDB.getMaxResponse(i); //get the maximum response
            //associate with this question.
            for(j = 1; j < max_response; j++) {

                expected_question_utility[i] = expected_question_utility[i] +
                getWeightedExpectedResponseUtility(j, i, trucks, rDB);
                //the WeightedExpectedResponseUtility is
                //P(Response) * ExpectUtilityOf(Response)
            }
        }
    }

    int biggest_question_id = 0;
    for(j = 1; j < rDB.numberOfQuestions; j++) {
        //search for the question with biggest expected utility;
        if(expected_question_utility[j] > expected_question_utility[biggest_question_id])
            biggest_question_id = j;
    }

    return biggest_question_id;
    //return the question with biggest expected utility.
}
```

43
//this method returns Probability of Response times the expected utility of this Response
private double getWeightedExpectedResponseUtility(int r, int q, Truck[] trucks, ResponsesDB rDB) {

    int cached_response = rDB.responseArray[q].response;
    rDB.responseArray[q].response = r;
    //Temporarily set the answer of the unanswered question q
    //to be the response r.

    trucks = makeBestBayesianGuess(trucks, rDB);
    //This procedure does the Bayesian Analysis
    //which computes the utility and all the
    //updated probabilities.

    double probability_of_response = getJointProb(q);

    double expected_response_utility = 0;
    int t; //temp variable
    //truck index.
    for(t = 1; t<trucks.length; t++) {
        //sum up all the weighted utility of all the
        //trucks.
        expected_response_utility = expected_response_utility +
            trucks[t].prioriProb * trucks[t].utility;
        //at this stage, since makeBestBayesianGuess is executed
        //the prioriProbs are actually the probability of the
        //truck purchase given the new response.
    }

    rDB.responseArray[q].response = cached_response;
    //restore the original response, which should be
    //the value that means not answered yet.

    return probability_of_response * expected_response_utility;
}