

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

Concordia Online Shopping Assistant:

**An Intelligent User Interface
For
Distributed Database Computing**

Guang Yu Li

**A Major Report
In
The Department
Of
Computer Science**

**Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada**

May 1999

©Guang Yu Li, 1999



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

395 Wellington Street
Ottawa ON K1A 0N4
Canada

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-43666-7

Canada

Abstract

Concordia Online Shopping Assistant: An Intelligent User Interface for Distributed Database Computing

Guang Yu Li

This project demonstrates a prototype for an intelligent user interface for an Online Shopping Assistant coded using Visual Basic 6, Oracle, and MS SAPI SDK on Windows NT environment. The purpose of this project is to explore state of the art technology to build a simple, functionally elegant, intelligent user interface to make software easy to use. This project has some unique features which differentiate it from other online shopping tools. The speech enabled GUI can response to user action accordingly and can give comments and recommendations to the user about a particular product. It has three language versions, English, French, and Chinese. Speech is only in English because the Microsoft speech engine has only an English version. It supports heterogeneous databases and has hyperlink features to link the image catalogue to a particular web site. The goal of this project is make the catalogue style online shopping enjoyable and retain the attention of potential consumers.

This report presents the architectural design of the COSA system, the design and implementation of client part GUI and server part databases. The success of most software is determined by its GUI, COSA system is no exemption. When designing and implementing COSA, a great deal of effort is focused on its usability and usability testing. Finally, future work related to COSA is presented.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Desai, for his kind agreement with my initial motivation and proposal. Subsequently his enthusiastic support and valuable guidance gave me an excellent chance to explore state of the art technology for this project. Without his kind support I could not finish it in time.

Second I would like to express my sincere thanks to my roommate, Sebastien Senechal for his help in French for this project.

Finally, I would like to dedicate this project to my wife, Ying Luo, for her full support, great patience, and encouragement. Without her financial support, I could not continue my graduate studies at Concordia University. This project is my great gift for her!

Subset of this project report entitled "Electronic Shopping Assistant" has been accepted by International Symposium on Electronic Commerce & Second International Workshop on Technological Challenges of Electronic Commerce sponsored by IBM, and to be presented on May 20th, 1999 in Beijing, People's Republic of China

Contents

1. Introduction	1
1.1 Overview of Electronic Commerce	1
1.2 Point and Click, Buy and Sell	2
1.3 COSA System	2
1.3.1 Introduction	2
1.3.2 Hypermedia/Multimedia	3
1.3.3 COSA: The Project	3
1.4 Organization of the Report	4
2. System Architecture of the COSA System	6
2.1 Intelligent Graphical User Interface	6
2.1.1 Task Analysis versus Requirement Specification	6
2.1.2 General Descriptions of Requirement Specification	6
2.1.3. Overview of Microsoft Visual Basic	7
2.1.3.1 What is Visual Basic?	7
2.1.3.2 Visual Basic vs. Object-oriented (OO) programming	8
2.1.4 Introducing Microsoft Speech Engine	9
2.1.4.1 Introduction	9
2.1.4.2 What is SAPI?	11
2.2 System Design	11
2.2.1 Client/Server Computing	11
2.4.2 System architecture	13
1) Input/Query Subsystem	14
2) Image/Agent Subsystem	15
3) Decision Subsystem	15
4) Order/Validation Subsystem	16
5) Database Subsystem	17
3. Input/Query Subsystem	18

4. Image/Agent Subsystem	22
4.1 Microsoft Data Access Model	22
4.2 Making Data Source Accessible	24
4.3 COM (Component Object Model)	25
4.4 ADO vs. DAO	26
4.5 Using ADO	28
4.6 Using DAO	28
4.7 Image/Agent Windows in COSA	29
5. Decision Subsystem	32
6. Order/Validation Subsystem	35
6.1 Placing an order	35
6.2 Secure Payment	37
6.2.1 What is ActiveX Component?	38
6.2.2 Implementing In-Process and Out-Of-Process Servers	39
6.2.3 Encrypt/Decrypt Process	40
6.2.4 Developing the Crypto Class	40
7 Database Subsystem	45
7.1 Database design	45
7.2 Customer Order Form	47
8. Usability Testing	48
8.1 What is Usability?	48
8.2 Usability Testing	48
8.3 Empirical Testing	49
8.4 Modification of GUI based on usability testing	52
9. Summary and Future Work	54
9.1 Summary	54

9.2 Future Work	55
Bibliography	57
Appendix	
User Manual	58
A. Server side Oracle database setup	58
B. Client Machine configuration	62
C. Run this program	65
D. Business Data Input Form	65
E. Software needed to run COSA system	67
F. Information about source files	67
F1. Oracle folder	67
F2. Client folder	68
F3. Crypto folder	70

List of Figures

1. TTS run-time synthesis	11
2. Systems Architecture of COSA	14
3. Decision Subsystem Flowchart	16
4. Main Window of COSA	18
5. Warning Dialogue	19
6. Welcome Window	19
7. Input Window	21
8. Different Data Access Model in VB 6	24
9. DAO Object Hierarchy	27
10. DAO Object Hierarchy	27
11. Image Window	29
12. Agent Window	29
13. Total Window	32
14. Change Window	32
15. Byebye Window	34
16. Order Window	35
17. Thanks Window	37
18. Encrypt/Decrypt Process	40
19. Reference to Crypto.dll component	42
20. Oracle SQL *Plus window shows the encrypted card number	43
21. Customer Order Form	47
22. Empirical Usability Testing instructions	50
23. Previous Change Window	53
24. Client-server configuration	56

1. Introduction

1.1 Overview of Electronic Commerce

The Internet is a gold mine of dynamic information. People are using it to quickly and easily research topics, including finance, travel and entertainment. With technological advances in security, the Web has also become a viable channel to buy and sell products and services. Internet companies such as Amazon, Yahoo, and Excite as well as telecommunication and cable companies recognize the huge number of customers and their buying power.

Netscape and AOL merged in a US\$4.2 billion deal in November 1998, bringing together Netscape's browser and electronic commerce (e-commerce) software business. In January, Excite and @Home (a high-speed Internet provider with reported 200,000 customers) struck a US\$6.7 billion agreement. Most recently, Yahoo bought GeoCities for US\$3.56 billion.

According to the latest forecasts from Forrester Research Inc. and International Data Corp. (IDC), e-commerce is expected to continue its rapid growth over the next few years. Forrester predicts that business-to-business e-commerce will grow from US\$43 billion in 1998 to US\$1.3 trillion in 2003—an increase of more than 3,000 percent. It also forecasts that business-to-consumer e-commerce will grow from US\$4.8 billion in 1998 to US\$17.4 billion in 2001—an increase of more than 360 percent. Alternatively, IDC predicts business-to-consumer e-commerce will grow from US\$10 billion in 1998 to US\$39 billion in 2001—a 390 percent increase. [Com99]

1.2 Point and click, buy and sell

E-commerce means doing financial transactions on the Internet, at an online store or over a secure business-to-business extranet.

E-commerce is online shopping, both wholesale and retail. Every day, more people gain access to the Web, and every day, more of them are shopping online. It provides a level of convenience they want, need and will soon demand. E-commerce is a unique opportunity for businesses of any size. And for those who automate their supply chain, the opportunity for business-to-business e-commerce is even greater.

The items most frequently purchased online were computer software, computers and peripherals, and books. Other popular items were music, financial products, travel, tickets for events and clothing. [Com99]

Currently it seems that electronic commerce has outgrown a single definition. It is now so far-reaching in scope, capability and purpose that it is part of every sector of the commercial world—from manufacturing to banking to distribution to retail operations. It is the future—and it is arriving fast. In this work, we focus on online shopping.

1.3 The COSA System

1.3.1 Introduction

When one looks at “virtual shopping mall” on the Internet such as the online pioneers Amazon.com, one finds that the interface in these sites is very primitive due to the

limitations of web browsers. One also notices that it has no true multimedia features. To clarify we give following definitions for hypermedia and multimedia.

1.3.2 Hypermedia/Multimedia

Hypertext is a text that contains links to other texts. The term was invented by Ted Nelson around 1965. [SK95] Hypertext is therefore usually non-linear.

HyperMedia is not constrained to be text-based. It can include other media, e.g., graphics, images, and especially the continuous media -- sound and video. Apparently, Ted Nelson was also the first to use this term. The World Wide Web (WWW) is the best example of hypermedia applications.

Multimedia means that computer information can be represented through audio, video, and animation in addition to traditional media (i.e., text, graphics drawings, images). Hypermedia can be considered as one of the multimedia applications.

1.3.3 COSA: The Project

The motivation of *Concordia Online Shopping Assistant* project is that the “catalogue style” shopping should be more enjoyable and should get the attention—also termed “mind share”—of the demographically attractive online consumer. But how can we achieve this goal? The answer comes from the smart and elegant GUI (Graphical User

Interface). This project involves client/server computing. Therefore, the client side user interface should be easy to use and should have multimedia and intelligence features to let the user have fun. These features are enhanced by state of the art technology from Microsoft, namely Visual Basic 6, speech engine, and third-party components running on Windows 95/98 and Windows NT.

COSA has the following unique features:

- *Speech Enabled GUI* (text-to-speech) the GUI can communicate with the user in English using speech as well as text.
- *Shopping Advisor* can give comments and recommendations to the user about a particular product.
- *Multiple Language Versions* the GUI supports three languages, English, French, and Chinese (GB).
- *Heterogeneous Databases* the GUI can access any ODBC data source (without modification) such as Oracle, DB2, MS SQL Server, Sybase, ISAM databases (dBase, FoxPro, Paradox, etc.)
- *Hyperlink* user can click on catalogue images leading to a particular web site, e.g., the producer of the product.

1.4 Organization of the Report

This report describes the architecture of the COSA system, and the design and implementation of the front-end GUI systems, and the back-end Oracle database systems.

Section 2 details the system design, the powerful Visual Basic programming language, Oracle, and Microsoft Speech Engine. The COSA system consists of five subsystems, which are described below:

The Input/Query Subsystem is presented in Section 3. In this section, five data access models available in Visual Basic 6 are described. Image/Agent Subsystem is described in section 4. Decision Subsystem is given in section 5. Order/Validation Subsystem is described in section 6. Database Subsystem is given in section 7. Usability testing is described in section 8. Finally a brief summary and future work are given in section 9. The detailed User Manual is also given in the appendix.

2. System Architecture of the COSA System

2.1 Intelligent Graphical User Interface

2.1.1 Task Analysis versus Requirement Specification

Task analysis in developing requirement specification encourages designers to choose from a range of options to optimize the design features for a novel interface. It is a prerequisite for GUI design. While task analysis is a useful tool for identifying the requirements of an interface, it does not in itself provide the design options that might best meet these requirements.

I follow the real world metaphor to design a GUI for COSA. For example, when a shopper comes to the store, he picks up a shopping cart, then browses the potential items he wants to buy, selects one or more items and puts them into the shopping cart. If he changes his mind, he can take out some items from his shopping cart, etc. The scenario is the same as the one used in the COSA. If user has experienced shopping, then he should have no problem to use COSA.

2.1.2 General Descriptions of Requirement Specification/Implementation

The COSA software (Driver Software, or DS) helps a user in online shopping an item quickly and easily. The DS will ask the user to do some queries, and show the catalogue image of the item on an **Image Window**. The user can browse the images one by one while an **Agent Window** pops up automatically and gives comments on each item shown

in the **Image Window**. The user can also make the **Agent Window** appear or disappear. The user can order one or more items, which will be charged to his/her credit card.

I take full advantage of Visual Basic 6.0's powerful features (e.g., MS Masked Edit Control 5.0, MS FlexGrid Control 5.0, MS Internet Control, etc.) and third party components (ActiveX controls such as Sheridan 3D, ActiveX32) to implement the interface. For example, MS Masked Edit Control can validate user input, i.e., only allow user to enter numbers not letters to improve the user's productivity, while MS Internet Control can easily link catalogue images to a particular web site. I also developed my own ActiveX component for the security issues. Particularly I use MS SAPI (Speech Application Programming Interface) Speech SDK 4.0 Suite to make the interface more attractive.

2.1.3. Overview of Microsoft Visual Basic

2.1.3.1 What is Visual Basic?

VISUAL BASIC 6 is THE NEWEST version of Microsoft's rapid application-development tool. Visual Basic is the fastest and easiest way to develop Windows applications. So what is Visual Basic? The "Visual" part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add pre-built objects into place on screen. Visual Basic provides an intuitive screen painter used to paint windows with a variety of controls (entry fields, check boxes, list boxes, push

buttons, and so on). Each control has properties (characteristics such as background color and caption, etc.)

Visual Basic also provides access to dynamic link library (DLL) functions written in such languages such as C, C++, and COBOL as well as access to object linking and embedding (OLE) calls to enhance Visual Basic applications.

Since the introduction of version 2, Visual Basic has provided database support. In version 6, Visual Basic has taken database connectivity and support to a new level. In the following sections, we will examine a few data access models used in this project.

Finally, Visual Basic is an event-driven programming language. It processes code not only in a simple serial fashion as do traditional programs but also as triggering events occur. Depending on the program's design, the program can immediately suspend normal execution and launch the code that is currently associated with a mouse click. Following this interruption, normal execution can resume.

2.1.3.2 Visual Basic vs. Object–Oriented (OO) Programming

Over the past decade object orientation and GUI have become the trend in the software development universe. As it turns out, windows (GUI) application development is best addressed using languages that refer to graphical objects rather than mere fields and records in a control block. Windows macros and controls exist in class libraries (another OO construct) and use messaging to communicate with one another and with the end

user. Many of the terms and concepts of GUI programming have their roots in OO technology, e.g., properties, methods, etc. Thus, a natural synergy exists between OO and GUI. Visual Basic has built in the capability to include a large degree of object orientation into the application development environment. In this project I built one Crypto class based on object orientation. Often in Visual Basic, an object is called a component.

In Visual Basic, a component can be a control, which is an externally supplied extension of the language (usually known as a VBX, OCX, or DLL), or it can be another application connected through OLE automation.

As we know, Java is a modern object-oriented language. Most things in Java are classes, which are inherited from the root class *Object*; the primitive numeric, character, and boolean types are the only exceptions. Without inheritance, you can not even build Java program. So now you can see the difference between component-oriented vs. object-oriented programming. In this point we can easily figure out VB is not object-oriented programming language, but it does support it.

2.1.4 Introducing Microsoft Speech Engine

2.1.4.1 Introduction

Speech in this project is limited to English because current version of the Microsoft speech engine only supports English. Traditionally, Text-to-Speech (TTS) systems convert input text into voice by using a set of manually derived rules for prosody

generation and/or voice synthesis. While these systems can achieve a high level of intelligibility, they typically sound unnatural. The process of deriving these rules is not only labor intensive but also difficult to generalize to a new language, a new voice, or a new speech style.

For speech generation, there are two main methods used: formant synthesis and concatenative synthesis. In recent years, data-driven approaches such as concatenative synthesis have achieved a higher degree of naturalness. Nevertheless, these speech units are still tediously extracted by human experts. As there are thousands of possible articulation contextual variations, the process of creating a good quality TTS system often takes years. Formant synthesizers may sound smoother than concatenative synthesizers because they do not suffer from the distortion encountered at the concatenation point.

Another data-driven approach used to minimize the number of concatenation points is to select large units, such as syllables or words. While this approach allows for excellent voice quality, it results in a large non-scaleable system, and it does not generalize well to new acoustic contexts.

The Microsoft Text-to-Speech engine is a concatenative synthesis engine, which means the audio output generated by the engine is generated from files, which contain information derived from recordings of real people. The output may be generated to

target use with a telephone or speakers attached to a computer. This TTS system is called *Whistler, Whisper Highly Intelligent Stochastic TaLkER*. [XAJH96] It is trainable, scaleable and natural. The naturalness of the speech is demonstrated in my project. The run-time text-to-speech synthesis diagram is given below. (see Figure 1)

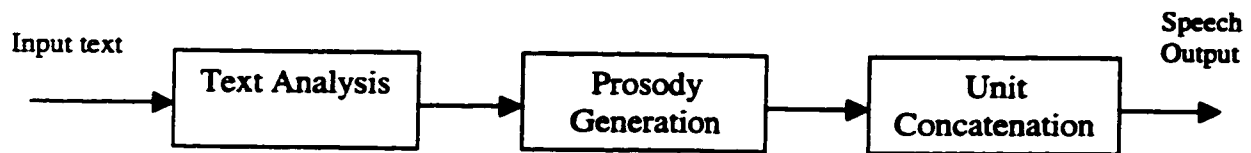


Figure 1 TTS run-time synthesis

2.1.4.2 What is SAPI?

SAPI stands for 'Speech Application Programming Interface'. It provides an API abstraction layer between applications & speech technology engines (both text-to-speech & speech recognition). This allows multiple applications to share speech resources on a computer and avoid the need for writing specialized application code for a specific speech technology engine. SAPI is the most widely supported speech API in use today.

In this project, I use the SAPI 4.0a SDK (Software Development Kits) just released on February 23, 1999 to develop speech enabled (text-to-speech) COSA system.

2.2 System Design

COSA is a two-tiered client/server application. Programs run on both the client and server computers.

2.2.1 Client/Server Computing

Client/Server computing is the logical extension of modular programming. Modular-based programming assumes that the separation of a large piece of software into its

constituent parts, or modules, creates easier development and better maintainability. Client/Server computing takes modular-based programming a step further by recognizing that the modules do not all need to be executed within the same memory space. With this architecture, the calling module becomes the client, and the called module becomes the server.

The basic characteristics of client server architectures are the following:

- Combination of a client that interacts with the user and server that interacts with the shared resource. The client process contains solution-specific logic and provides the interface between the user and the rest of the application system. The server process acts as a software engine that manages shared resources, such as databases, printers, modems, or high-powered processors.
- The front-end task and back-end task have fundamentally different requirements for computing resources, such as processor speeds, memory, disk speeds and capacities, and input/output devices.
- The environment is typically heterogeneous and multivendor. The hardware platform and operating system of client and server are not usually the same. Client and server processes communicate through a well-defined set of standard application program interface (APIs).

In a distributed client/server database environment, the server is responsible for intelligently servicing a client's request for data. The workstation does not request data at a file or table level, but sends a request to the server to execute a query and return specific records. This is a vast improvement over the file-server approach.

Client/Server also allows us to add new processing power without recoding the application. Client/server applications should not be aware of the location of their data. As a result, if data has to be moved from one server to another, the application itself should not have to be modified to reflect the location change.

This translates into increased productivity. If the mainframe goes down, all processing stops. In a client/server environment, if one server goes down, only the users of that server are effected. And the effect is temporary since processing can be quickly moved to another machine. As we approach the age of 24-hour data access, this will become more and more important.

Client/server computing allows for the use of new 4GL technology to develop applications faster and better. Graphical design products, object-oriented products and others are the key to increased future productivity and are primarily available on a PC.

2.4.2 System Architecture

Based on the above client/server descriptions and user task analysis, I designed the architecture as follows (see Figure 2)

As shown here, the GUI system has five subsystems, Input/Query Subsystem, Image/Agent Subsystem, Decision Subsystem, Order/Validation Subsystem and Database Subsystem. Each of the first four subsystems has one or more interactive or responsive windows leading to a particular intention.

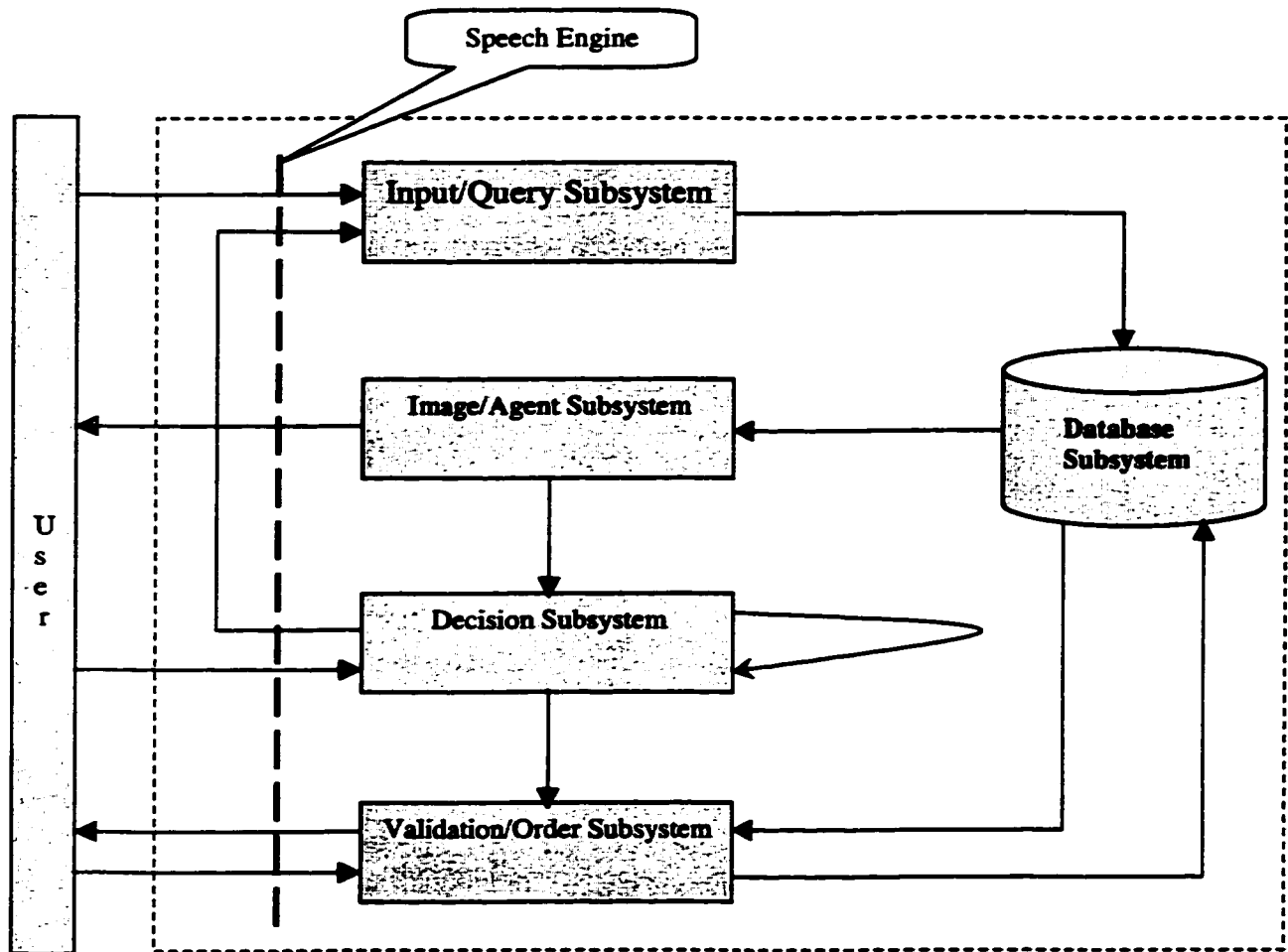


Figure 2 System Architecture of COSA

1) Input/Query Subsystem

Input Subsystem has three interactive windows: **Main Window**, **Welcome Window**, and **Input Window**. User can choose category catalogue information in **Main Window**, this

leads to the **Welcome Window** which prompts user to choose a preferred language, English, French, or Chinese. Next comes the **Input Window**, user can check particular boxes in order to browse the online catalogue.

2) **Image/Agent Subsystem**

This subsystem has two windows: **Image Window** and **Agent Window**. In the **Image Window**, the user can browse the images of products back and forth, and put (select) one or more items into his/her “shopping cart”. Here I follow the real world shopping metaphor to design the interface. Meanwhile, the **Agent window** pops up automatically and shows comments and suggestions accordingly. Users can make the **Agent Window** appear or disappear from the **Image Window**. The user can also go back to the **Input Subsystem** for another query as he/she wishes. Or, the user can go to the **Decision Subsystem**.

3) **Decision Subsystem**

This system has three windows: **Total Window**, **Change Window** and **Dialogue Window**. After the user has selected one or more items from **Image Window**, he/she enters the **Total Window**. Because this system is the core of Shopping Assistant, it is necessary to show the interaction between **Total Window** and the other two windows: (See Figure 3)

From the **Total Window**, the user has a number of choices, i.e. if the user changes his mind or the total amount is out of budget, he/she can go to the **Change Window** to cancel some or all the items in his/her shopping cart. When user confirms all the items

he/she selected or cancelled, he comes to the **Dialogue Window**. From this point, he/she can either go to **Order/Validation Subsystem** to pay by the credit card, or **Input/Query Subsystem** to make another purchase, or just quit **COSA system**.

4) Order/Validation Subsystem

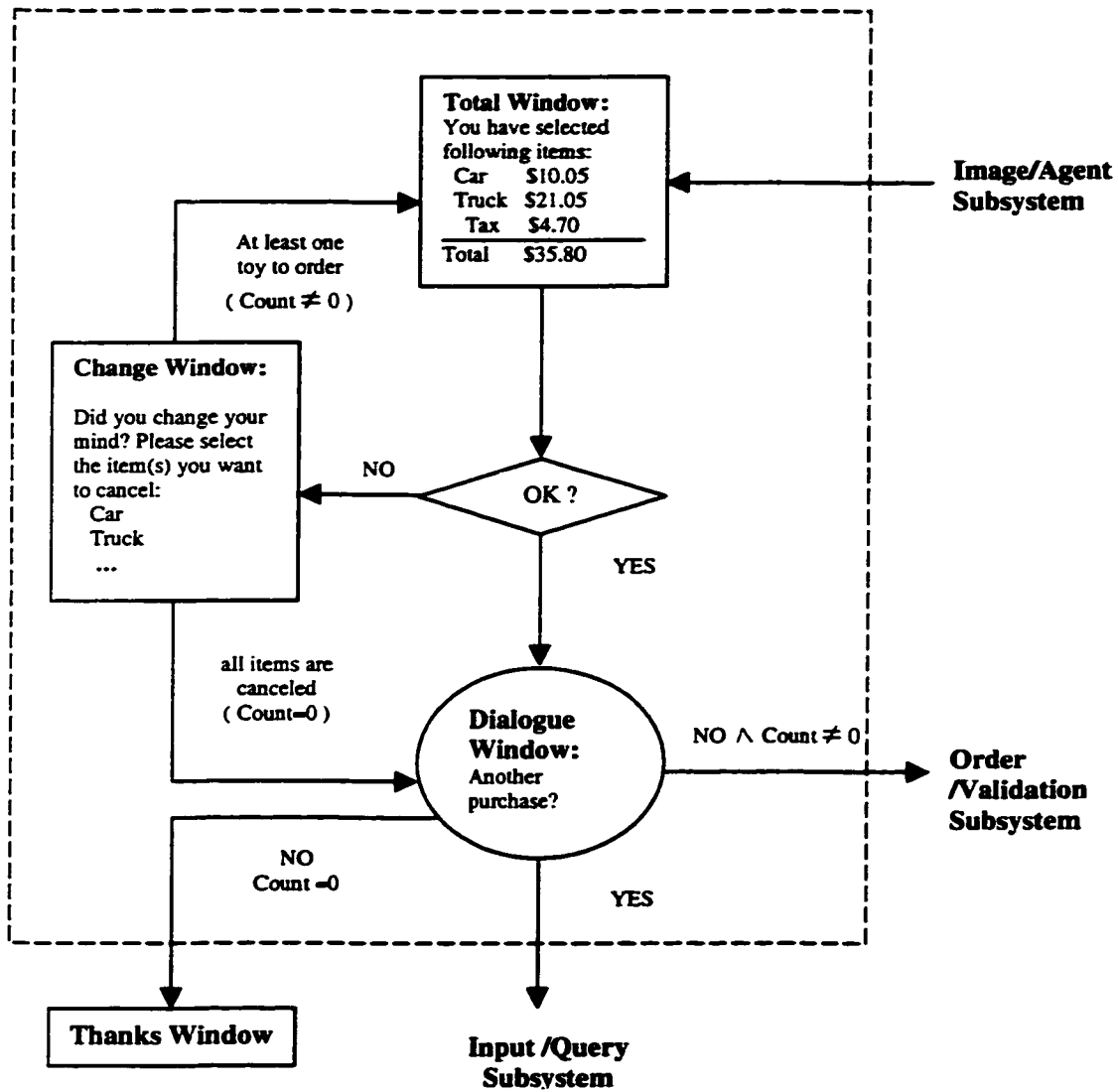


Figure 3 Decision Subsystem Flowchart

This subsystem has two windows: **Order Window** and **Thanks Window**. User uses his credit card to make a payment. **Order Window** prompts user to enter his card number. If the number is approved, the system goes to the **Thanks Window**. If number is not approved, a validation message box pops up. The validation message box informs the user of an invalid card number and prompts the user to reenter the card number. If the credit card has expired, the validation box informs the user about that. Here the credit card number is encrypted before being sent to the server database.

5) Database Subsystem

Because Oracle 8 supports a large number of user connections simultaneously, the current version of the database system is implemented using Oracle 8. It has **MY_CUSTOMER**, **TOY**, and **MY_ORDER** table in the database. There are three windows in this subsystem, **Customer Order Window**, this server-side window gives detailed information about the order requests from online shoppers. The credit card number is decrypted from the **MY_ORDER** table and is shown in this window. The **Catalogue Data Input Window** lets business update and browse catalogue information. **Customer Data Input Window** lets business input and browse customer information.

3. Input/Query Subsystem

When COSA system starts to run, the first window shows up.

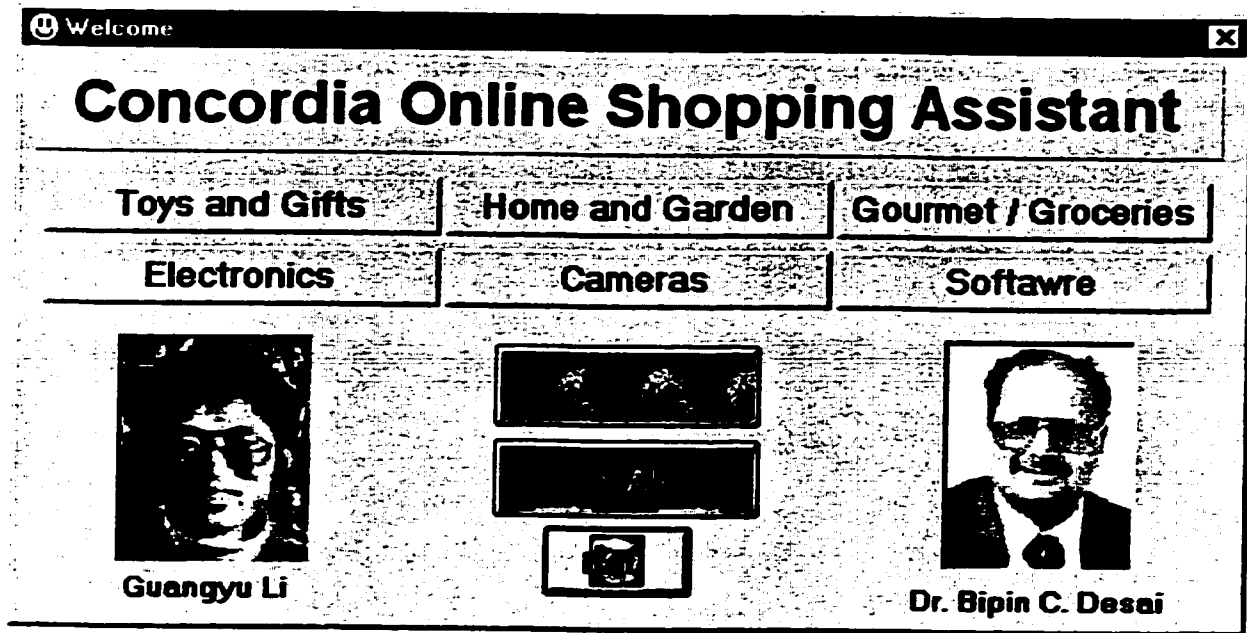


Figure 4 Main Window of COSA

From here, the user has many options, such as “Toys and Gifts”, “Electronics”, etc. He can select his preferred category. In our case, select “Toys and Gifts”. Other categories haven’t been implemented, but the scenario is the same. Because of this project’s distributed features, these categories or catalogue databases could be located anywhere on the Internet.

The user can also notice that there are two images in this window. If the user places the mouse pointer over these two images, the mouse pointer icon is changed indicating that the images are clickable objects. In this project, if the user places mouse pointer over the clickable controls in the window, the mouse pointer icon will be changed. This is an important feature in GUI design called “quick feedback”. For example, if the user clicks

on the image, then Microsoft Internet Explorer will open the corresponding author's homepage. If the user points to these images or other command button for one second, a tooltip pops up providing brief further information. Another good GUI design principle is forgive user's wrong action. For example, if the user clicks on the Exit button or on the "close icon" located at the upper right corner on this window, COSA system will open a dialogue box to ask the user to confirm that action, to avoid accidental termination of this program. (see Figure 5) Throughout this project these are the fundamental design rules in my GUI implementation.

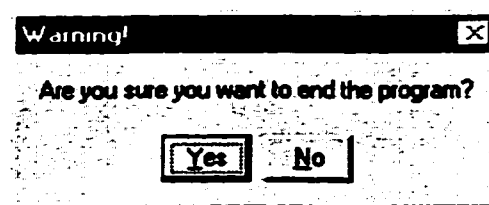


Figure 5 Warning Dialogue

Next if the user clicks on the "Toys and Gifts" button, this leads to the "Virtual Toy Store". Then the welcome window shows up, in the meantime, the Main window is closed. (See Figure 6)

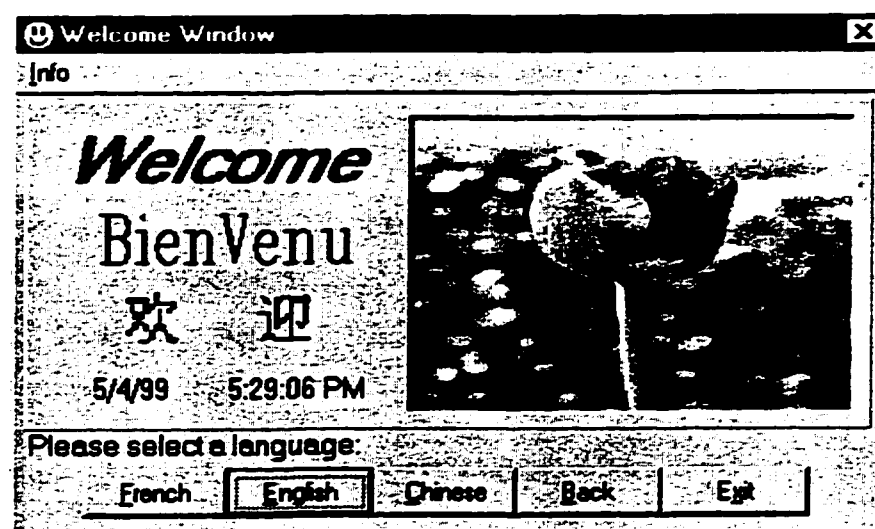


Figure 6 Welcome Window

When this window starts up, the message “Please select a language:” moves in from the right hand side. This animation is controlled by the Timer control in VB, it is hidden from the user. You can also see the date and time on this window. In this window, the user can select preferred language service, English, French, or Chinese (GB). The Chinese version doesn’t support *Big 5* character which is widely used in Hong Kong. User can also click on “Back” button to go back to Main Window to select other categories. This give users more flexibility! Of course, the user can also terminate the program at this point.

In Visual Basic, localizing software to a particular language is easy. First construct a resource file (.res file), which consists of strings and string IDs (similar to the hash table containing key/value pair in other programming language such as Perl and Java). Then use **LoadResString(index)** VB built-in function to load a string on the form. Here *index* is an integer specifying the identifier (ID) of the data in the resource file. For example, in my project, I assign English ID as 10, Chinese as 100, and French as 300.

Using **LoadResString** is very useful for localizing a Visual Basic application because the resources that need to be translated are isolated in one resource file and there is no need to access the source code or recompile the application.

After the user chooses one language, the **Input Window** opens (See Figure 7). The user can ask about some toys suited for the checked age range. In this window there are some check boxes grouped by age range and special offer.

Notice that when the user clicks on “Continue” button, the client part interface tries to connect to backend Oracle database server through the network, and it will take some time depending on the network speed. The ProgressBar control shows up on the bottom of this window to provide feedback and inform user that “please wait, I am trying to connect to the database”. If user has no indication that an operation is taking place, he/she may assume that the application has stopped responding. Also, when users get feedback, they feel that application is taking less time than it actually is. The ProgressBar is a great tool to use for that reason.

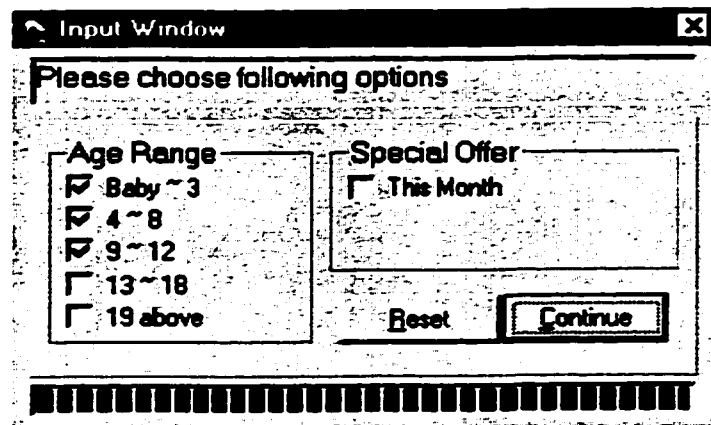
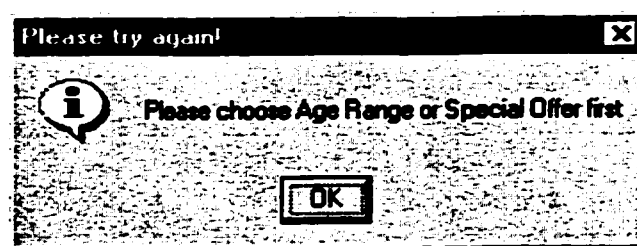


Figure 7 Input Window

If user did not choose the Age Range or Special Offer and attempts to click on Continue button, then a message box shows up (see following figure).



4. Image/Agent Subsystem

4.1 Data Access Model

The Image/Agent Subsystem uses database connections through Microsoft data access technology. There are several data access models available, as described below.

These models simplify the development of client/server data-enabled applications. Visual Basic 6 provides support for several data access models. Each model represents a different state of the art in the evolving data access technology.

Data Access Object (DAOs), which communicates with Microsoft Access and other ODBC-compliant data sources through the JET database engine. DAO is a native access model for Access.

Open Database Connectivity (ODBC), which is an API call interface to the Open Database Connectivity libraries and drivers to provide data access to Microsoft SQL Server and other databases that provide an ODBC driver.

Remote Data Objects (RDOs), which provide a framework for using code to create and manipulate components of a remote ODBC database system.

ODBCDirect, which access ODBC data sources through the RDO with DAO objects, bypassing the JET database engine.

ActiveX Data Objects (ADOs), which eliminate the need to choose DAO and RDO and any other data access method, and is designed to provide a common bridge among different databases, file systems, and e-mail servers. This is the latest data access model from Microsoft.

OLE DB

Visual Basic 6 provides full support for the OLE DB and ADO data access methodologies, but what are ADO and OLE DB all about? Microsoft introduced ODBC with the promise of creating a single common access methodology for databases. The earliest versions of ODBC suffered from inconsistent support and performance. In fact, ODBC was supported by very few database products, and those that did support ODBC also provided their own database drivers, which often were more reliable and faster. ODBC has come a long way from those early days. In its current form, two versions of ODBC are available-Version 2.0, which supports 16-bit applications, and, with the release of Windows 95, Version 3.0, which supports 32-bit applications. Today, almost every major database supports ODBC drivers, and third-party developers provide optimized driver versions. In fact, ODBC drivers have become as ubiquitous as video drivers and other Windows device drivers. Also many data processing applications (such as Excel, Access, Lotus 1-2-3, and so forth) support ODBC data access. ODBC has become the omnipresent methodology for providing access to database sources.

The primary focus of ODBC is to provide a consistent interface to database data sources.

OLE DB is designed with an even broader goal in mind: to provide a methodology to

access data, regardless of the data source. OLE DB becomes the data access bridge for documents, e-mail systems, file systems, spread sheets, COM components, and other database sources that utilize ODBC drivers.

Figure 8 summarizes these access models.

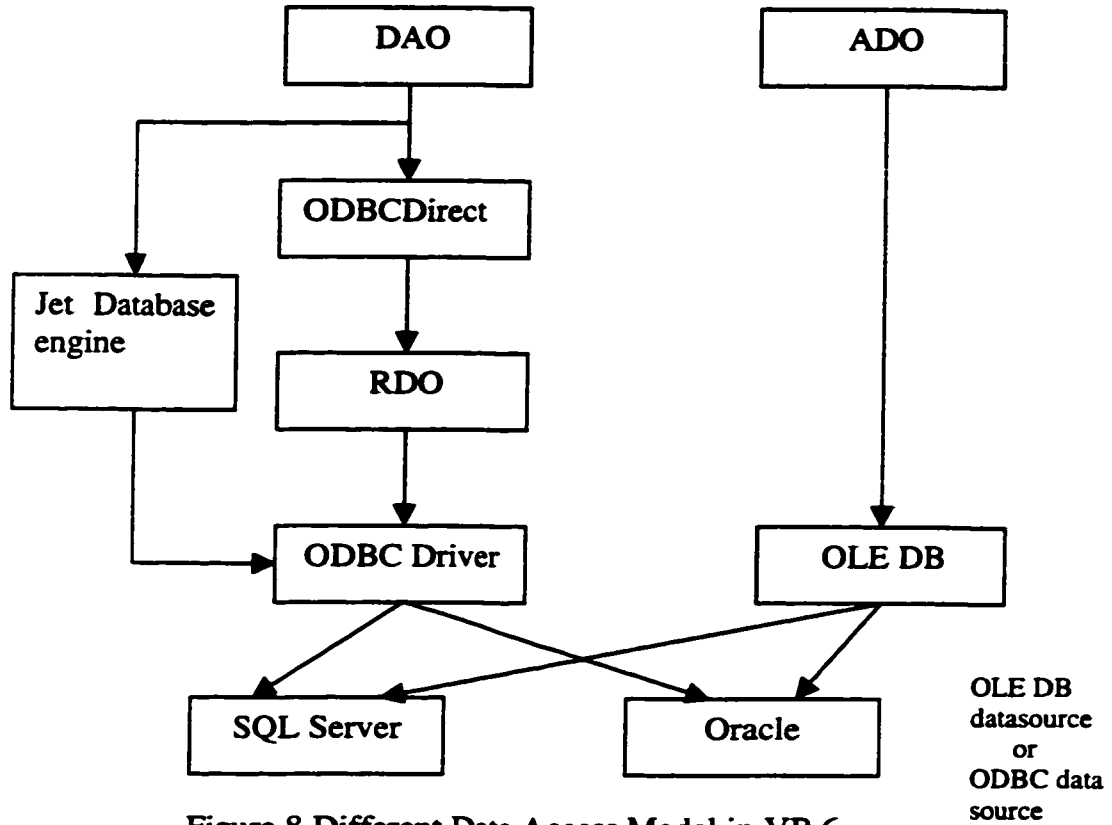


Figure 8 Different Data Access Model in VB 6

4.2 Making Data Sources Accessible

The key feature of OLE DB is that it lowers the requirements for implementing a data provider interface. Previously, to provide an ODBC interface, an application was required to implement a database engine that was capable of interpreting and executing SQL queries. With OLE DB, a data provider is not required to support a command interface. In

conjunction with a query processor, OLE DB provides a unified way to access enterprise data.

How is OLE DB different from ODBC connectivity? OLE DB extends the capabilities of ODBC, by providing the capability for less-sophisticated data applications to become data providers. This doesn't mean that ODBC interface will be abandoned, however. The ODBC Provider allows ADO to connect to any ODBC data source. ODBC will still be used to support database data sources, but instead of relying on the ODBC interface, applications will utilize the OLE DB interface to access these data sources.

4.3 Component Object Model (COM)

The idea of component-based software is a hot topic in the last two years. Microsoft has been working on developing this component-based model of application development for a long time. First came OLE (Object Linking and Embedding), which facilitated interapplication communication by enabling application objects to be embedded in other applications. For example, a spreadsheet object could be placed in a word processing document, and a user was allowed to edit that object in place. Visual Basic Controls (VBXs) were introduced with Visual Basic applications. These controls could be Windows interface elements, such as list boxes and buttons. VBXs could also provide specialized processing routines, such as specialized parsing and sorting routines. OLE and VBXs were Microsoft's first steps toward developing object-oriented application components. OLE and VBX were later merged into OLE Controls (OCXs, which are OLE-based custom controls). After OCX, the technology evolved into ActiveX, the key

difference of which is that ActiveX controls are designed to function cross-platform. Microsoft has ported ActiveX to the Macintosh environment, and Sun has licensed ActiveX with an eye toward porting it to its architecture.

COM is specification for developing application components that can dynamically exchange data and that can be interchanged to support new functionality. COM employs object-oriented programming techniques to build encapsulated application components. These components provide an interface to an object. This interface is used to manipulate the object's state, and can be determined dynamically at run time. The COM architecture makes the development process an independent task by enabling components to work together, even if they have been developed with different programming languages and by different people. COM also provides the capability for components to work together in a distributed environment. [JD99]

4.4 ADO vs. DAO

ADO is an API for developing applications that can access OLE DB data providers. ADO is supported in various different programming languages, including Visual C++, VBScript, Visual J++, and Visual Basic, as well as in Active Server Pages. While using OLE DB directly provides a very low-level approach to accessing OLE DB providers, ADO provides a higher-level and easier-to-understand mechanism. ADO is a combination of the best aspects of DAO and RDO and doesn't rely as heavily on the object hierarchy as DAO and RDO. This means that manipulating and creating ADO objects is much easier, because they can be created and managed directly.

To see how different the DAO and ADO class hierarchy are, let's look at the following figures first.

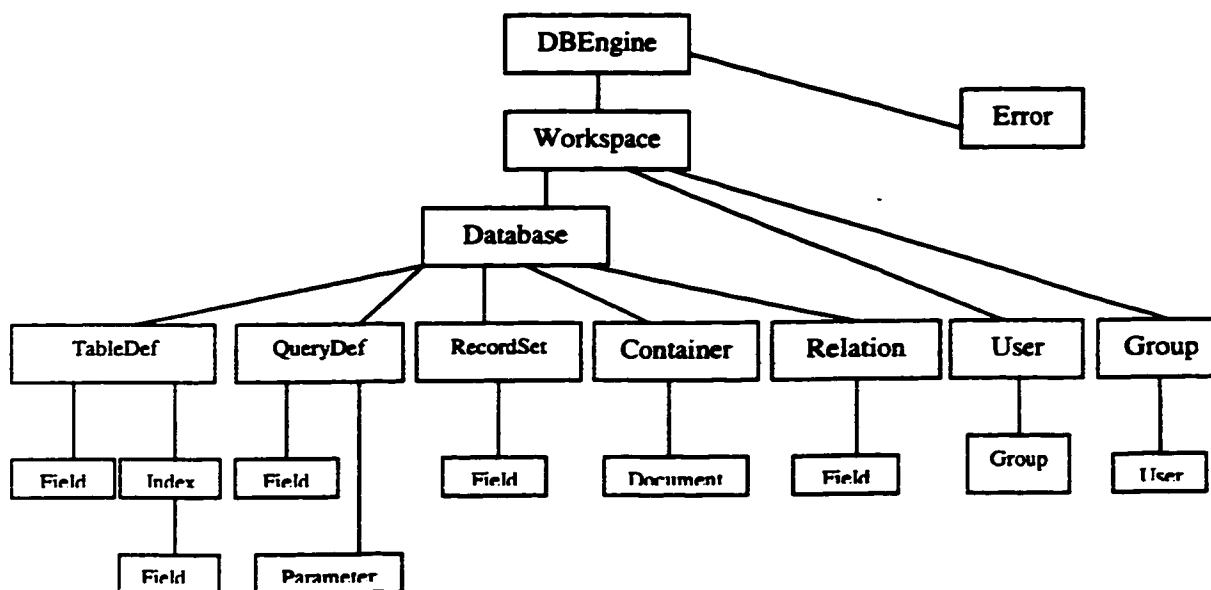


Figure 9 DAO Object Hierarchy

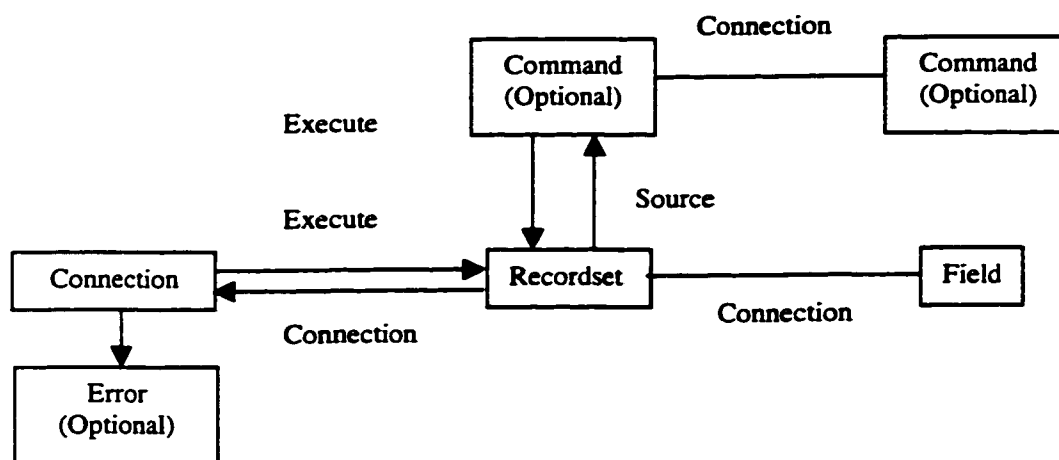


Figure 10 ADO Object Hierarchy

From the DAO object hierarchy diagram, we know that we have to create a set of objects such as DBEngine, Workspace, Database, Recordset, etc. to establish connection and get data from database. While for ADO, we can use only one object such as Recordset to

establish connection and get the data we want. The following code example from this project illustrates how to use these models to access a back end Oracle database.

4.5 Using ADO

Using ADO to connect to an ODBC data source is much easier than using DAO, the earliest data access model in Visual Basic. Following is the example for how to connect to an Oracle database using a Recordset object.

```
Dim rs As ADODB.Recordset
Set rs = New ADODB.Recordset
rs.Open "MY_ORDER", "DSN=myoracle8;UID=g_li;PWD=g_licindi;"
On Error Resume Next
```

4.2.4 Using DAO

Using DAO to connect to database needs a little bit more effort. You have to follow the DAO class hierarchy to establish the connection. You can not omit any object (two objects in this program, db and rs.) Following is the DAO code to connect to Oracle database.

```
Public db As Database ' the database object
Public rs As Recordset
Set db = DBEngine.OpenDatabase("myoracle8", False, False, _
    "ODBC;DSN=mylocation8;UID=g_li;PWD=g_licindi;")
Set rs = db.OpenRecordset("MY_ORDER", dbOpenDynaset)
On Error Resume Next
```

4.7 Image/Agent window in COSA

There are two windows in this Image/Agent subsystem, i.e., Image Window (See Figure 11) and Agent Window (see Figure 12). After the user issues the query in the Input/Query

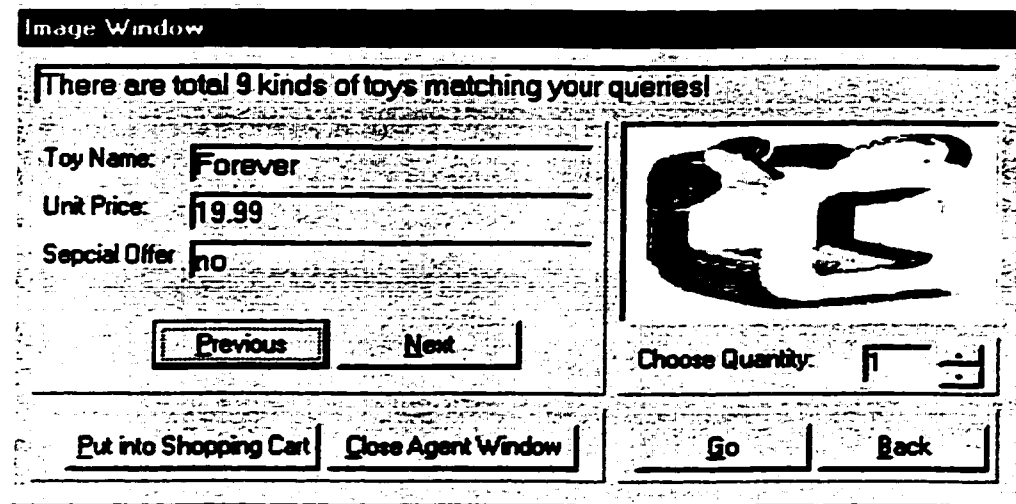


Figure 11 Image Window

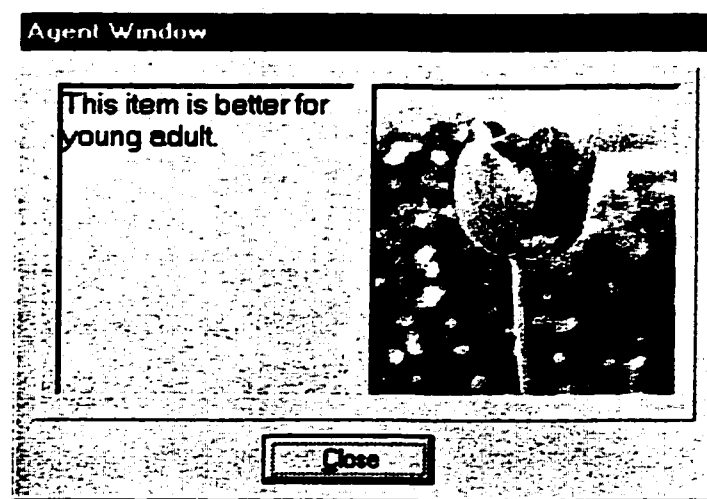
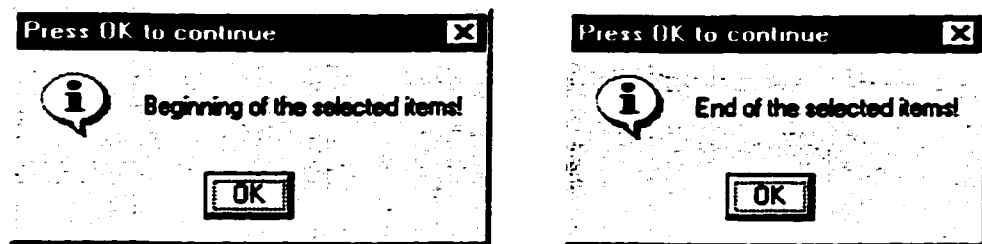


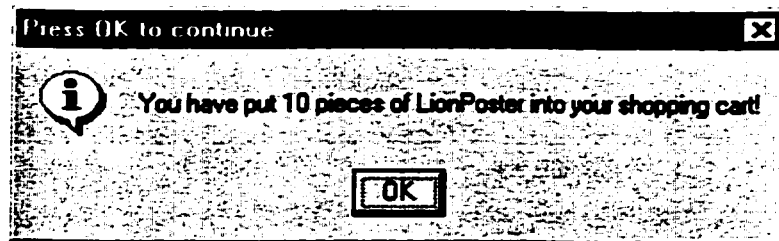
Figure 12 Agent Window

subsystem and the database connection is successful, these two windows show up simultaneously. The Image Window is located at left hand side on the screen, the Agent Window on the right. If you place the mouse pointer over the some controls such as the image and command button, the mouse icon is changed. For example, the user can click

on the image, then Microsoft Internet Explorer will open a particular web site, say the producer of this product. This is also indicated by the tool tip if user place the mouse over the image one second. On the upper part there is a label message indicating how many items match the user's query. The user can browse the selected items back and forth using Previous and Next button, while the Agent Window shows comments and suggestions accordingly. If user reaches the first or the last item, a message box pops up the following figure.



The user can set quantity very easily by using scroll bar just below the image. The user can close the Agent Window by clicking on the button labeled "Close Agent Window", the button label is changed to "Open Agent Window" while the Agent Window is closed. Or alternatively, the user can click on the "Close" button on the Agent Window. If the user puts some items in his shopping cart by clicking "Put into Shopping Cart" button, a message box will confirm this action, see the following figure:



If user did not put anything into his shopping cart and tries to go to next step, i.e., click on the “Go” button, then another message box shows up:



In the Image Window, the user can also click “Back” button to go back to Input Window to do another query. If the user is satisfied with his selection, then he can click “Go” button, which comes to the Total Window in Decision Subsystem.

5. Decision Subsystem

There are two major windows in this subsystem, Total Window and Change Window. In Total Window (see figure 14) a detailed summary is presented to the user, i.e., Toy Name, Unit Price, Quantity, Subtotal, GST/PST, and Total amount. So, the user can have an idea if he can afford these items.

Total Window

In your shopping cart you have

Toy Name	Unit Price	Quantity	Subtotal	GST/PST
Bike	\$79.99	1	\$79.99	\$12.45
Camera	\$34.9	5	\$174.5	\$27.15
Flower	\$5.99	12	\$71.88	\$11.18
Realearth	\$7.99	16	\$127.84	\$19.89
Pat the Bunny	\$9.99	11	\$109.89	\$17.1
Forever	\$19.99	8	\$159.92	\$24.88
Lamaze Soft Stacking Rings	\$1000.	10	\$10000.	\$1556.
			Total:	\$12392.67

Do you want to order what you have selected?

Yes No Exit

Figure 13 Total Window

Change Window

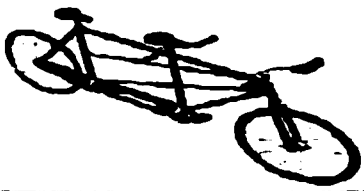
There are total 1 items in your cart.

Toy Name: Bike

Unit Price: 79.99

Quantity: 1

Previous Next



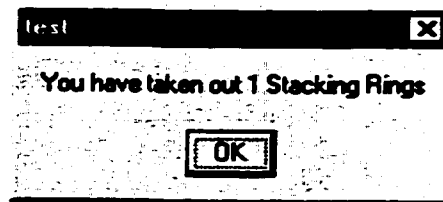
Quantity: 1

Take Out Go Exit

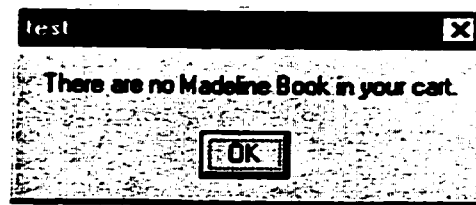
Figure 14 Change Window

If the user thinks that is too expensive or due to some other reasons, he can cancel these items by clicking on the “No” button. The **Change Window** shows up. See Figure 14. In the **Change Window**, the user can browse the items in his shopping cart. He can set the number of items to be taken out by using the scroll bar. Notice that the user can not set the number greater than the current corresponding number of item in the shopping cart. This greatly improved the user’s productivity while online shopping.

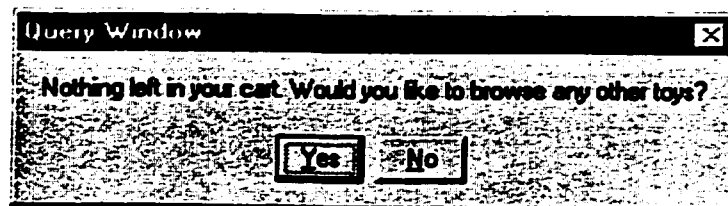
After user set the number to be taken out, he clicks on the “Taken Out” button, a confirmation message will be presented as the following:



If no particular item left in the shopping cart, and user attempts to continue to take out, another message box shows up:



If nothing left in the shopping cart, and user click on the “Go” button, COSA gives the following message:



If user click “No”, this will terminate program. COSA presents ByeBye Window. See Figure 15. If click on “Yes”, user will go back to Input Window to do another query.

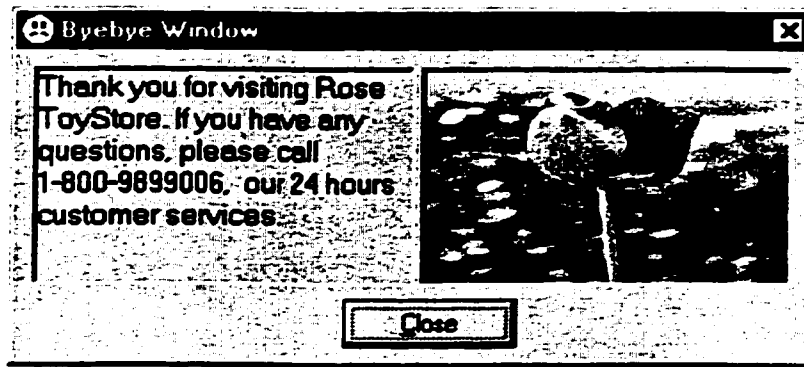
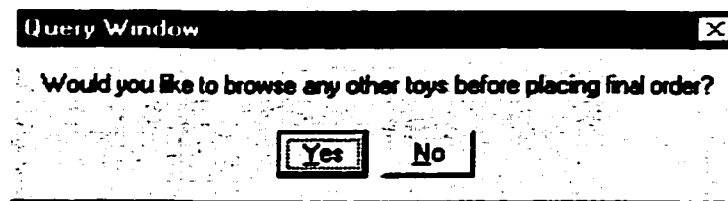


Figure 15 Byebye Window

From the Total Window, if the user clicks on “Yes” button, the dialogue window shows up, see the following window:



If the user clicks on “Yes” button, this will lead the user back to the Input Window (see Figure 7) to do another catalogue browsing. This give the user another chance before placing final order. If the user clicks on “No” button, he enters the Order/Validation subsystem, which is given in the next section.

6. Order/Validation Subsystem

6.1 Placing an Order

If the user is satisfied with what he has put into his shopping cart, he wants to place an order on them. Subsequently he enters the Order/Validation Subsystem. There is one major window in this subsystem. See Figure 16.

Order Window

Your Shipping Address

Name: []

Apt. No: []

Street: []

City: []

Province: []

Zip Code: []

Phone No: []

Select Your Credit Card

Visa Card

MasterCard

American Express

Enter Your Card Number

[]

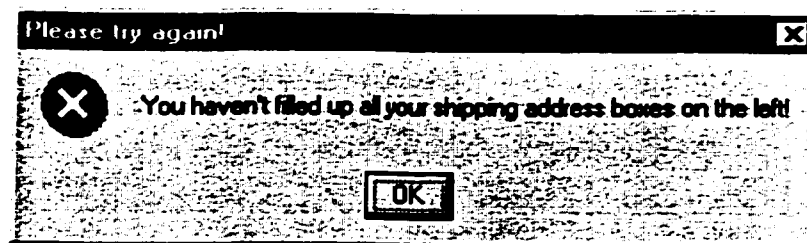
Expiry Date

[]

Reset Go Exit

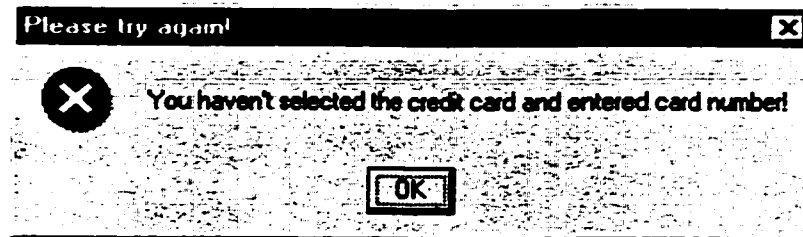
Figure 16 Order Window

If the user leaves everything empty in this window and click on “Go” button, the system will respond with the following message:

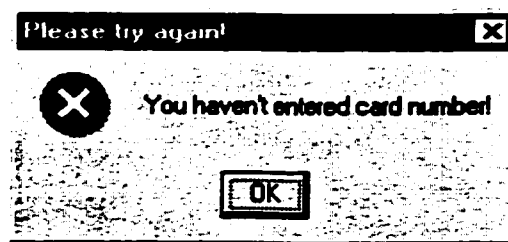


In the shipping address Phone No. input box, user can only input numbers. If try to input other characters, these characters will not show up in this box. Nothing appears! This also improves user's validation.

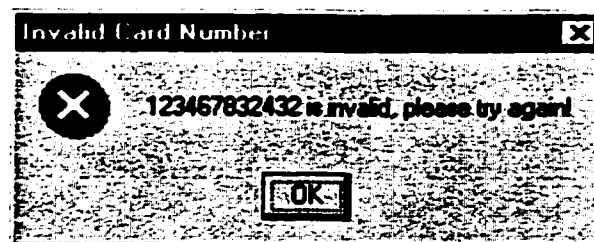
If the user filled up all the address boxes, then immediately click on the "Go" button, then system presents a message:



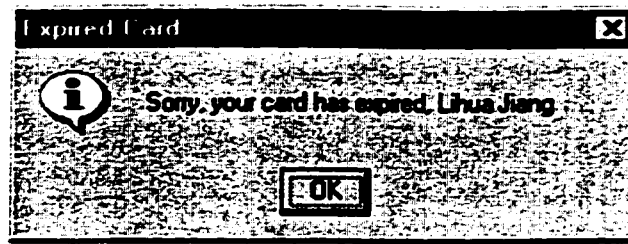
If the user selected a credit card, but didn't input card number and click on the "Go" button, then system gives another message:



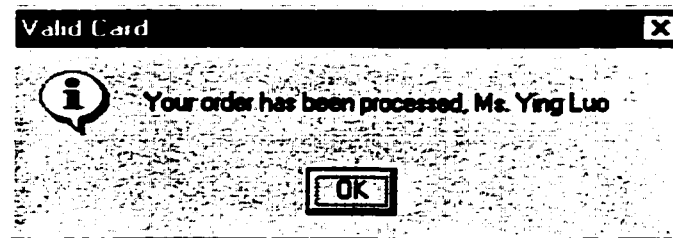
If the user inputs an invalid card number, the message will be presented to the user:



If the user's credit card has expired, then the system will present the message:



If the user inputs the correct card number and the card has not expired, then system gives following message:



If user click on OK on this message window, then Thanks Window will shows up.

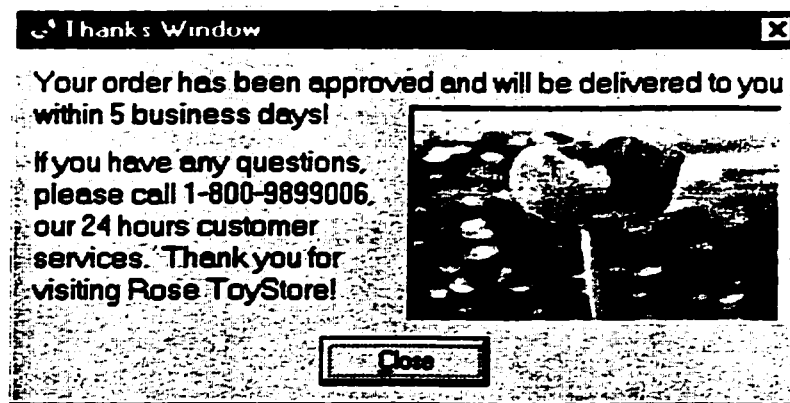


Figure 17 Thanks Window

6.2 Secure Payment

In order to make online shopper feel secure and safe, we have to address the secure payment issues. In this project, I designed and implemented an ActiveX code component to solve this problem.

6.2.1 What is ActiveX component?

An ActiveX component can be any of the following:

- A code component
- An ActiveX control
- An ActiveX document

Code components are server applications that expose their functionality through an interface consisting of properties, methods, and events. The difference between ActiveX controls and code components is that ActiveX controls have a visible interface and are integrated into the Visual Basic IDE. Code components are classes that can be accessed through properly declared object variables with the `CreateObject()` function.

ActiveX documents are applications that can be hosted in containers such as Internet Explorer and the Office Binder. At this time, there aren't many ActiveX documents on the Internet, and it seems it will be a while to catch up.

ActiveX code components provide a functionality similar to that of ActiveX controls, but they are not as integrated with the development environment (for example, you can't drop a code component on a Form as you can a control), and they don't have a visible interface. Instead, they must be accessed in the same way that built-in objects are accessed with the object variables. ActiveX components are implemented as classes. They are prototypes that can be used to create objects. Classes exhibit the two most important benefits of object-oriented programming: Abstraction and Reusability .

6.2.2. Implementing In-Process and Out-of-Process Servers

A class module is a server—an application that provides its services to the client application. When you create an object variable to access the properties and methods of a class, you are actually invoking an executable file (DLL or EXE) that runs in the background and waits to be contacted. Every time you set or read a property value or call a method, this executable is activated, it performs some action, and, optionally, returns some result to your application.

You can implement servers as ActiveX DLL or ActiveX EXE components. The difference between the two is how the server is executed. An ActiveX DLL is an in-process server. The DLL is loaded in the same address space as the executable that calls the server, and it runs on the same thread as the client. At any given moment, however, either the client application or the DLL is running. The benefit of DLLs is that they are faster, because, in effect, they become part of the application that uses them.

An out-of-process server runs as a separate process. When a client application creates an object provided by an EXE server for the first time, the server starts running as a separate process. If another client application creates the same object, this object is provided by the running EXE server. In other words, a single EXE server can service multiple clients. Out-of-process servers seems to be more efficient in terms of resource allocation, but exchanging information between servers is a slow process. Therefore, in terms of execution speed, in-process servers are faster. In this project, I developed an ActiveX DLL component.

6.2.3 Encrypt/Decrypt Process

Once a message has been encrypted, it can be stored on nonsecure media or transmitted over a nonsecure network and still remain secret. Later, the message can be decrypted into its original form. This process is shown in the following illustration.

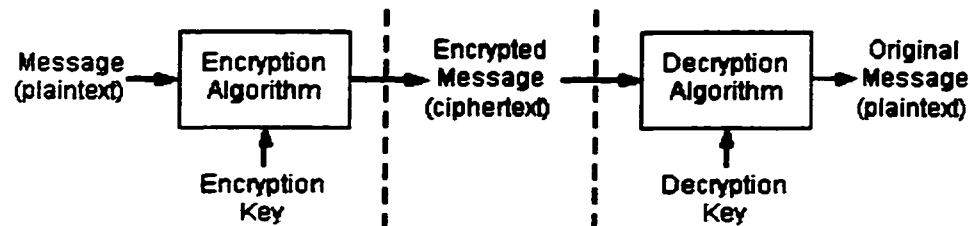


Figure 18 Encrypt/Decrypt Process

Data encryption and decryption is a simple process. When a message is encrypted, an *encryption key* is used. This is comparable to a key that is used to lock a padlock. To decrypt the message, a *decryption key* must be used. The encryption and decryption keys are often, but not always, the same key.

It is very important to keep the keys safe and transmit them securely to other users. However, the challenge is to properly restrict access to the decryption key, because anyone who possesses it will be able to decrypt all messages that were encrypted with the corresponding encryption key.

6.2.4 Developing the Crypto Class

The cryptography engine used by the Crypto class is fairly simple, yet it will prevent most occasional intruders. The encryption technique is based on the XOR operator, which has the following unique property: If you XOR a character with another character, you will

get an encrypted version of the original character. If you XOR this character with the same key character once again, you will get the original one. The Crypto class source code is the following:

```
'Option Explicit
'local variable(s) to hold property value(s)
Private mvarText As String
Private mvarKey As String
Private mvarEncryptedText As String

Public Function Encrypttext() As Integer
Dim textChar As String * 1
Dim keyChar As String * 1
Dim encryptedChar As Integer

    If mvarText = "" Then
        Err.Raise vbObjectError + 100, "Crypto.CryptoClass", _
            "Can't encrypt null text"
        Encrypt = 0
        Exit Function
    End If
    If mvarKey = "" Then
        Err.Raise vbObjectError + 101, "Crypto.CryptoClass", _
            "Encryption key not specified"
        Encrypt = 0
        Exit Function
    End If

    mvarEncryptedText = ""

    For i = 1 To Len(mvarText)
        textChar = Mid(mvarText, i, 1)
        keyChar = Mid(mvarKey, (i Mod Len(mvarKey)) + 1)
        encryptedChar = Asc(textChar) Xor Asc(keyChar)
        mvarEncryptedText = mvarEncryptedText &
            Chr(encryptedChar)
    Next
    Encrypt = 1
End Function

Public Property Get EncryptedText() As String
    EncryptedText = mvarEncryptedText
End Property

Public Property Let Key(ByVal vData As String)
    mvarKey = vData
End Property
```

```

Public Property Get Key() As String
    Key = mvarKey
End Property

Public Property Let Text(ByVal vData As String)
    mvarText = vData
End Property

Public Property Get Text() As String
    Text = mvarText
End Property

```

We can make a Crypto.dll file from the VB environment and put this file into Windows NT or 95/98 system directory, then we register this component by issuing this command under dos command line (two files in the same directory): regsvr32.exe Crypto.dll

Then we can refer this component in the project. See Figure 19.

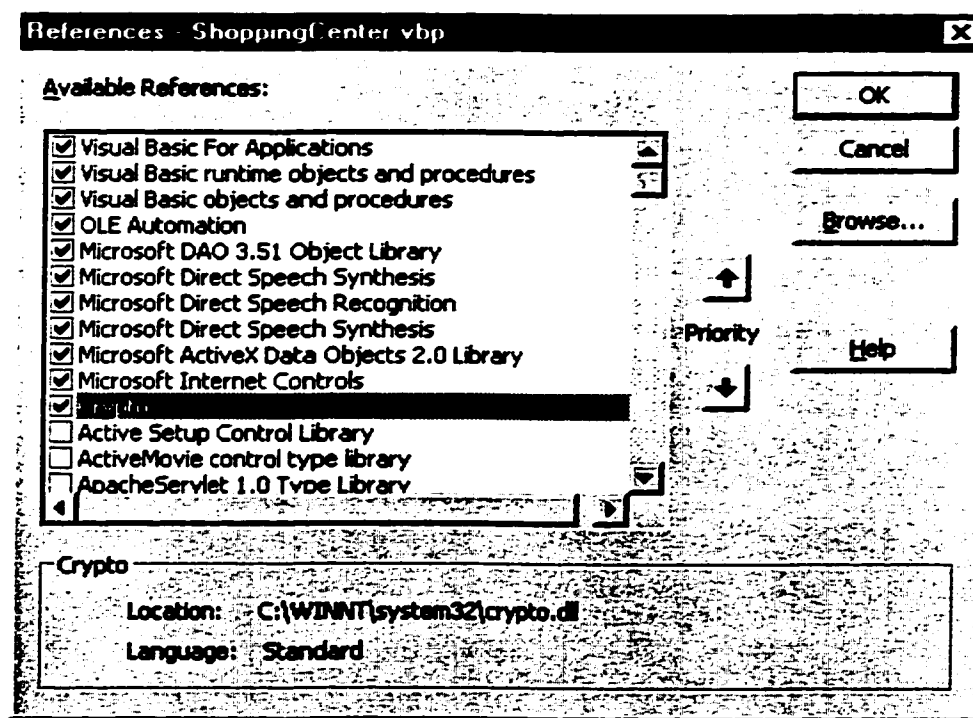


Figure 19 Reference to Crypto.dll component

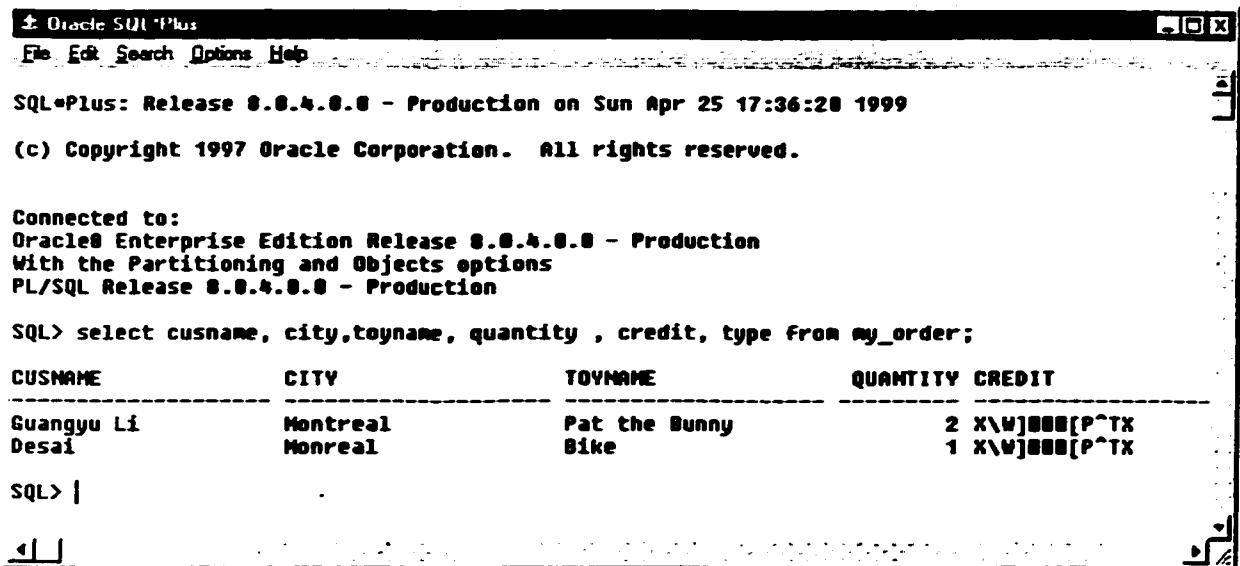
After we make a reference to this component, we can use it the same way as any other component. For example, we can create an object called `CryptoObj` as the following:

```

Private CryptoObj As New CryptoClass
Private secretText As String
'encrypt the credit card number contained in mskBox.Text
CryptoObj.Key = "cindi928"
CryptoObj.Text = mskBox.Text
Call CryptoObj.Encrypttext
secretText = CryptoObj.EncryptedText

```

If the card number is 123456789001 then the encrypted string stored in the Oracle database will look something like the following figure:



The screenshot shows the Oracle SQL Plus interface. The title bar reads "Oracle SQL Plus". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The main window displays the following text:

```

SQL*Plus: Release 8.0.4.0.0 - Production on Sun Apr 25 17:36:20 1999
(c) Copyright 1997 Oracle Corporation. All rights reserved.

Connected to:
Oracle® Enterprise Edition Release 8.0.4.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.0.4.0.0 - Production

SQL> select cusname, city,toyname, quantity , credit, type from my_order;

```

CUSNAME	CITY	TOYNAME	QUANTITY	CREDIT
Guangyu Li	Montreal	Pat the Bunny	2	X\W]000[P^TX
Desai	Monreal	Bike	1	X\W]000[P^TX

The screenshot also shows the prompt "SQL> |" at the bottom left of the window.

Figure 20 Oracle SQL Plus window shows the encrypted card number

To decrypt the card number, we have the following code (the decryption window will be presented in Section 7 Database Subsystem):

```

'decryption
CryptoObj.Key = "cindi928"
CryptoObj.Text = rs.Fields("CREDIT")
Call CryptoObj.Encrypttext
txtCreditNumber.Text = CryptoObj.EncryptedText

```

The encryption engine I implemented for the Crypto class isn't terribly secure. The problem with this scheme is that it uses a fixed-length key. If the original text contains a pattern of asterisks, dashes, or other special characters, the length of the key will become

evident. One simple trick to make this encryption scheme more secure is to blend the key with the text as you move along. When you run out of characters in the key, you wrap the key and start with its first character again. But you can replace it with a new key, based on the text characters you've encrypted already. For example, instead of using the first character of the key for a second time, you can XOR its first character with the first character of the text and use the result of the operation as the first character of the key. This process is reversible, because when you run out of characters in the encryption key, the first few characters of the original text will be available.

7. Database Subsystem

7.1 Database Design

There are three tables in this project database, MY_CUSTOMER, TOY, and MY_ORDER. Each of them is described below.

MY_CUSTOMER has following fields:

```
CREATE TABLE MY_CUSTOMER (
    CUTID          VARCHAR2 ( 10) NOT NULL,
    CUTNAME       VARCHAR2 ( 20)  NULL,
    CDTYPE        NUMBER ( 3, 0) NOT NULL,
    CDNO          VARCHAR2 ( 20) NOT NULL,
    EXDATE        DATE           NOT NULL
) ;
```

CUTID customer id;

CUTNAME customer name;

CDTYPE credit card type;

CDNO credit card number;

EXDATE credit card expire date.

TOY table has following fields:

```
CREATE TABLE TOY (
    TOYID          NUMBER ( 11, 0) NOT NULL,
    TOYNAME        VARCHAR2 ( 50) NOT NULL,
    AGERANGE       NUMBER ( 3, 0) NOT NULL,
    UNITPRICE      NUMBER ( 12, 2) NOT NULL,
    SPECIAL        VARCHAR2 ( 50)  NULL,
    TOYIMAGEPATH   VARCHAR2 ( 50) NOT NULL,
    CHINESE        VARCHAR2 ( 255) NULL,
    FRENCH         VARCHAR2 ( 255) NULL,
    NOTES          VARCHAR2 ( 255) NULL,
    URL            VARCHAR2 ( 255) NULL
) ;
```

TOYID toy id;
TOYNAME toy name;
AGERANGE this toy is suitable for which age range;
UNITPRICE unit price;
SPECIAL whether this toy is special offer or not;
TOYIMAGEPATH image directory in the hard disk
CHINESE Chinese comments/suggestions
FRENCH French comments/suggestions
NOTES English comments/suggestions
URL producer web site address

MY_ORDER has following fields:

```

CREATE TABLE MY_ORDER (
  CUSNAME      VARCHAR2 ( 20) NOT NULL,
  APTNUMBER    VARCHAR2 ( 10) NOT NULL,
  STREET       VARCHAR2 ( 20) NOT NULL,
  CITY         VARCHAR2 ( 20) NOT NULL,
  PROVINCE     VARCHAR2 ( 20) NOT NULL,
  ZIPCODE      VARCHAR2 ( 15) NOT NULL,
  PHONE        VARCHAR2 ( 20) NOT NULL,
  TOYNAME      VARCHAR2 ( 20) NOT NULL,
  ORDERDATE    DATE,
  QUANTITY     Number NOT NULL,
  CREDIT       VARCHAR2 ( 20) NOT NULL,
  TYPE         VARCHAR2 ( 5) NOT NULL,
  EXPIRYDATE   DATE

```

);
CUSNAME customer name;

APTNUMBER apartment number;

STREET street name;

CITY city name;

PROVINCE geographical province;

ZIPCODE	postal code;
PHONE	telephone number;
TOYNAME	ordered toy name;
ORDERDATE	the date of order;
QUANTITY	number of ordered toy;
CREDIT	credit card number;
TYPE	credit card type.
EXPIRYDATE	expiry date of credit card.

7.2 Customer Order Form

This is the server side GUI to provide a convenient way to check the order placed by a customer for product delivery. See Figure 21.

The screenshot shows a window titled "Customer Order Form" with a close button in the top right corner. The window contains a form titled "Customer Order Info" with the following fields and values:

Name	Guangyu Li
Apt #	1805
Street	Mackay
City	Montreal
Province	Quebec
Zip Code	H2G 1H9
Phone #	51 49899006
Toy Name	Pat the Bunny
Order Date	05/04/99
Quantity	2
Credit Card #	123456789001
Card Type	AmericanExpress
Expiry Date	10/99

At the bottom of the window, there are three buttons: "Previous", "Next", and "Exit".

Figure 21 Customer Order Form

Note that the Credit Card number has been decrypted. See Figure 20 to compare.

8. Usability Testing

8.1 What is Usability?

Usability is clearly defined as “...convenient and capable for use...” [Rhi71]. The term usability to describe effectiveness of human performance was first used by Bennett (1979). In the following years a more formal definition was proposed by Shackel (1981) and modified by Bennett (1984). Finally Shackel (1991) simply defined usability as “the capability to be used by humans easily and effectively”, where

easily = to a specified level of
subjective assessment

effectively = to a specified level of
human performance [Wog97]

Usability is not a consideration but can be crucial to acceptance and success of a product in the marketplace. In recent years the technical areas of human factors engineering and ergonomics have focused a great deal of attention on the usability and ease of use of machines and systems by people to enhance human productivity. [Rhs85]

8.2 Usability Testing

Usability testing is a generic name for a set of methods based on having evaluators inspect or examine usability-related aspects of a user interface. The set of methods can be

Heuristic evaluation, Guideline reviews, Consistency inspections, Cognitive walkthroughs, etc. [RJ95] Usability inspectors can be usability specialists, end users with content or task knowledge, or other types of professionals. In this case, I use empirical evaluation for the usability testing.

The main goal of testing is to improve the COSA system user interface. It is concerned with making the systems easy to use and easy to learn. I focus a great deal on the following issues:

- The relevance of the system, how well it serves the users' needs
- The efficiency, how efficiently users can carry out their tasks using the system
- The users' attitude to the system, their subjective feelings
- The learnability of the system, how easy the system is to learn for initial use and how well the users remember how to use the system
- The safety of the system, giving the users the right to "undo" actions and not allowing the system to act in a destructive way, e.g., to cancel a shopping item without confirmation of the user.

8.3 Empirical Test

There are total of 5 different users who performed the testing. The testers were computer science students aged from 26 to 34 who had never seen the application before and were, therefore, total beginners. The users were given written instructions (see Figure 22) on paper and an opportunity to ask any questions they liked before the test started. When

the test had started, questions were not answered directly, instead they were answered with a question. For example, if the user asked “What do I do now?” the answer would be of a general type like “What do you think you could do?” No further help was given during the test.

Electronic Shopping Assistant (ESA)

ESA is a tool where you can get information about toys and make order for your own or others.

Please try to reach the four following goals:

- 1) You want to buy some toys for different people, one for a young adult, two for your friend’s baby, three for yourself and five for your friends.
- 2) After you know the total price is out of your budget, you decide to buy only two toys for your friend’s baby.
- 3) You changed your idea again after you know you can afford more than two toys, so you decide to select another two toys for yourself.
- 4) You make an order for the above four toys using your American Express credit card. Your card number is 123456789001.

Figure 22 Empirical Usability Testing instructions

The test consisted of four goals the user had to reach which were chosen in such a way to force the user to examine different features of the system.

To get the subjective feeling of the users about their experience with the use of the system, we produced a short questionnaire. The answers to it are also shown below.

- a. Did you have any problems during the test?
 - Did not get any feedback when moving the mouse

- b. Which part was the most difficult (if any), or the most irritating?
- Didn't get enough information about the contents in the shopping cart in order to take out some particular toys.
- c. Would you use this software (an improved version) in the future? Why, or why not?
- Yes, probably, it is useful because it make shopping much easier and more enjoyable.

An advantage of performing a user test is that you actually see the user using the application which can give you additional information about how to improve the user interface. The aim was also to see which task the user performed and the errors he made.

I observed the users during the tests of the system, what they did and what they said, and what problems they seemed to have. Based on this data, i.e., observations of the users during their use of the system, I tried to draw some conclusions as to how the user interface should be changed and improved. These empirical conclusions are based on the answers to the questions above and on the background knowledge of user interface evaluation. During the test I didn't give any time constraint to the users and the users did not receive any help from the observers.

The most important observations were:

After users put many toys into their shopping cart, and want to cancel some of them, they forgot which one should be taken out of their cart. The reason is in the "Change Window", I show the contents of the shopping cart in a "List Box" without corresponding images. The users could not remember all of them.

8.4 Modification of GUI based on usability testing

Therefore, an improved version of the system should include improvements on these issues. I decided to change the user interface as follows:

- Provide some feedback when moving the mouse over “clickable” objects
- Change some labels and messages, providing useful and helpful information
- Add toy images in the “Change Window” to visualize corresponding toys in the shopping cart, also add the scroll bar to speed up user’s action of cancellation.
- Make the font bigger.

This information did not change the user interface drastically. That is, I did not perform major changes, such as changing the structure of the program. Instead I made smaller changes that users will notice when they actually use the application. The most visible changes were made in the Change Window part (see Figure 14) compared with the previous one (See Figure 23). The changes can be seen in the image area and the removal of the “List Box”. I also place the image area on the right side of the Change Window because I know the controlling object should be put on the left hand side, the controlled object on the right side of the window.

User task analysis plays a key role in UI requirement specification. User-centered system design enhances software product usability. I conform the interface design to the Microsoft Windows 95 standard, so make it consistent with other MS Windows

applications. I also noticed that developing a good GUI system requires people good at design, writing, test and evaluation, and coding.

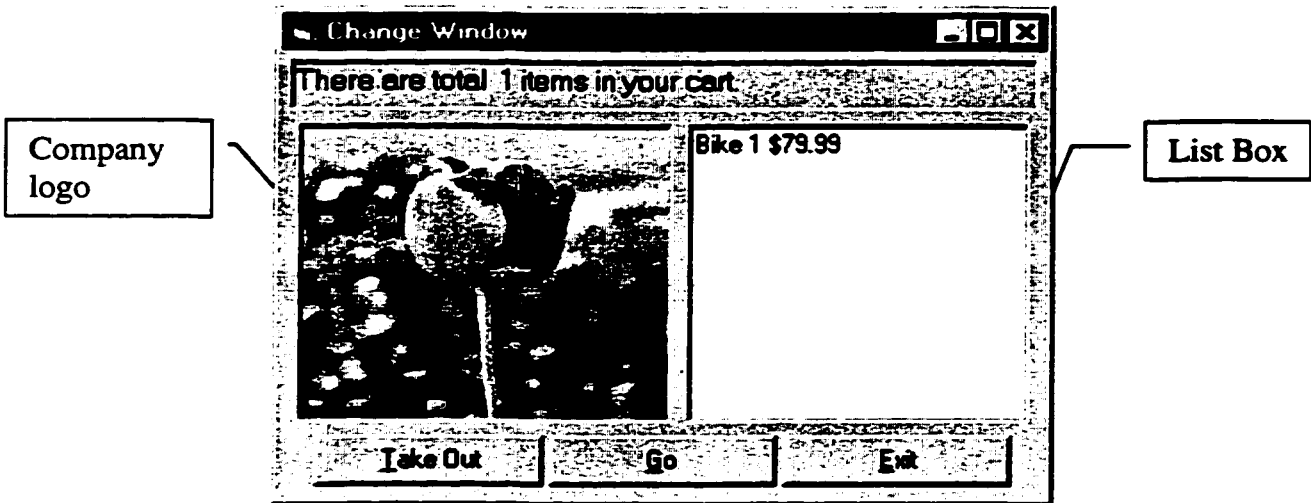


Figure 23 Previous Change Window

9. Summary and Future Work

9.1 Summary

The rapid development of the Internet is changing the way commerce is conducted, and is leading to economic globalization. Electronic commerce is becoming a major channel for conducting business, with more and more organizations developing, deploying and installing electronic commerce products, applications and solutions.

Concordia Online Shopping Assistant (COSA) is motivated by the idea that online shopping should be more enjoyable and get more attention of potential consumers. Without current state of the art technology available we could not achieve that goal. COSA takes advantage of the latest technology available from Microsoft Visual Basic 6, MS Speech SAPI 4.0a SDK (released in February 1999), and Oracle 8 Universal Database Systems.

The current version of COSA has some unique features, namely multimedia and intelligence. It can response to the user's action quickly and intelligently. It has three language versions, English, French and Chinese to serve majority in the world. This is especially useful for the multicultural environment in the Internet community. COSA also supports heterogeneous data sources such as ISAM data sources, ODBC data sources (Oracle, DB2, Sybase, MS SQL Server, etc.) to meet the diversity of available data on the Internet. It also has the distributed features such that the databases could be located anywhere on the Internet.

9.2 Future Work

The secure payment has been addressed for a long time in the E-commerce world and a dedicated web site has been set up at <http://www.semper.org/sirene/outsideworld/security.html#crytool>

The current version of COSA system has not addressed this issue considerably. Advanced encryption algorithm should be considered for use. PGP, Pretty Good Privacy, is a high security RSA public-key encryption application for MS-DOS, Unix, VAX/VMS, and other computers. PGP encrypts data using the International Data Encryption Algorithm with a random session key, and uses the RSA algorithm to encrypt the session key. So it is more secure than the algorithm used in COSA project. There is a real world example NetMarket (<http://www.netmarket.com/>) based on PGP and Mosaic/Netscape. Future work could focus on using PGP to make COSA more secure on payment.

COSA is a two-tiered client/server application. The current version has not been tested on physically separated client and server machines. The two-tiered client/server configuration diagram is given below (see Figure 24). Oracle's SQL*Net allow database clients to access a database remotely. SOL*Net is a multiprotocol proxy which forwards database requests from the current host to the host that has the data. This proxy component can be thought of as middle tier, sitting between the application and the databases the application uses.

The detailed information of configuration of both client and server machine is given in the user manual. Future work also can be done to test COSA on separate machines. The

networked NT machines (except the one in LB-928 on which I worked) in the Department of Computer Science at Concordia University has not been installed the necessary Oracle network software at this writing. So whenever you configure the client machine through the ODBC applet, you will get the following message:

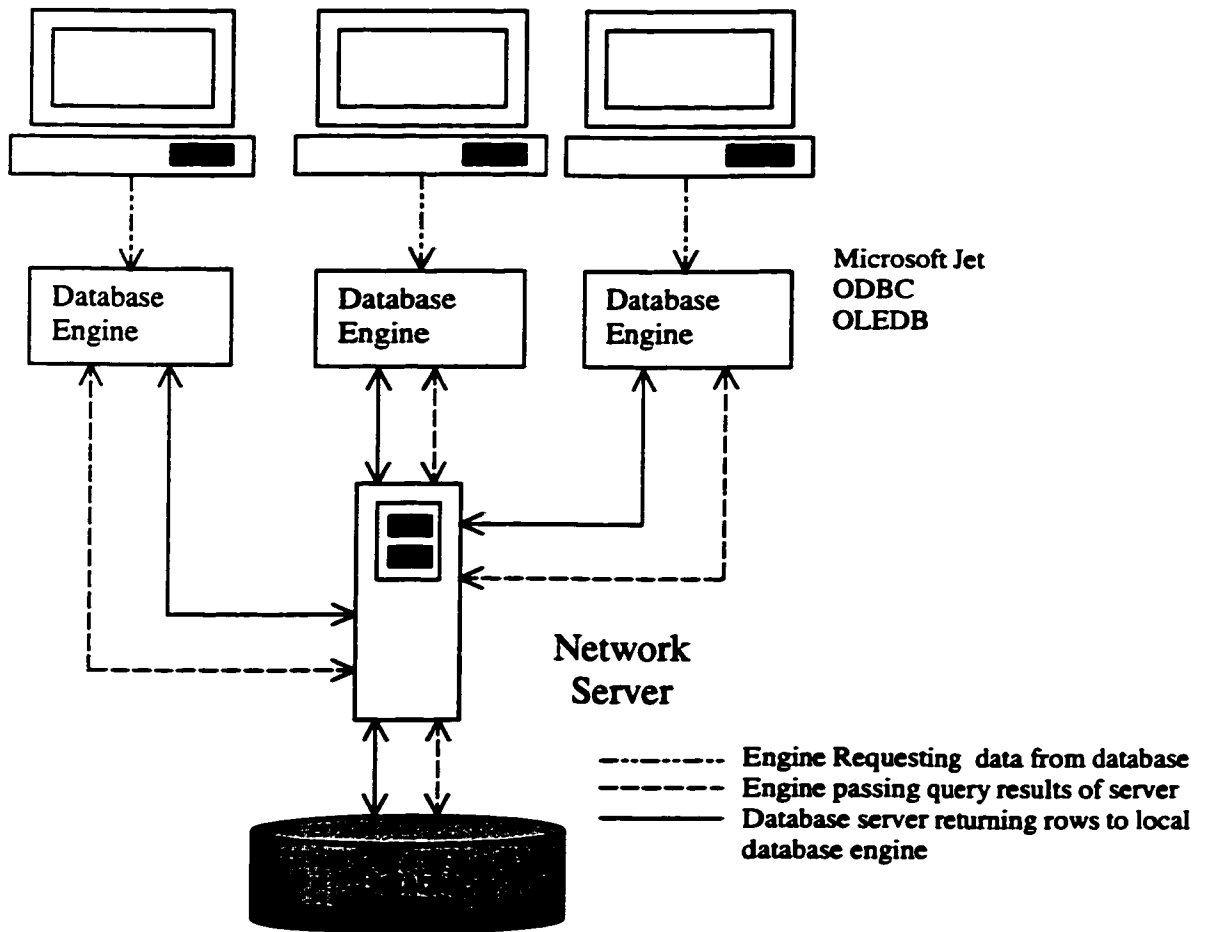
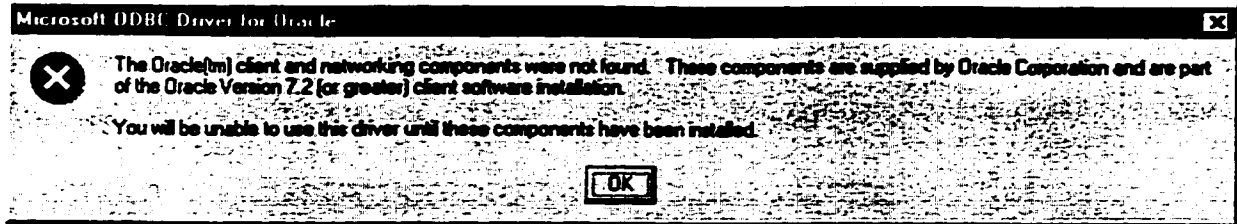


Figure 24 Client-server configuration

Bibliography

- [AF98] Andreas Lecerof, Fabio Paterno, Automatic Support for Usability Evaluation, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. 24, No.10, October, 1998.
- [Com99] Computer Paper, www.tcp.ca, Eastern(Montreal Area) Edition, May 1999, Volume 12, No.5
- [JD99] John W. Fronckowiak, David J. Helda, Visual Basic 6: Database Programming, IDG Books Worldwide, Inc., pp9, 1999
- [Jos95] Joseph B. Greene, Oracle DBA Survival Guide, Sams publishing, 1995.
- [Rhi71] The Random House Dictionary of the English Language, Random House Inc. pp.1573, 1971.
- [Rhs85] Richard H. Spencer, Computer Usability Testing and Evaluation, Prentice - Hall, Inc., pp.10, 1985.
- [RJ95] Robert L. Mack, Jakob Nielsen, Usability Inspection Methods: Executive Summary, Readings in Human-Computer Interaction: Toward the Year 2000, Morgan Kaufmann Publishers, Inc., pp.172, pp.176-179, 1995.
- [SK95] R. Steinmetz and K. Nahrstedt, "Multimedia: computing, communications & applications", *Prentice Hall PTR*, 1995.
- [Wog97] Wilbert O. Galitz, The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, John Wiley & Sons Inc., pp.87, 1997.
- [XAJH96] Xuedong Huang, Alex Acero, Jim Adcock, Hsiao-Wuen Hon, International Conference of Spoken Language Processing, Philadelphia, 1996.