

---

Student Work

---

11-1999

## To Host a Legacy System to the Web

Miguel Angel Bustamante  
*University of Nebraska at Omaha*

Follow this and additional works at: <https://digitalcommons.unomaha.edu/studentwork>

 Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Bustamante, Miguel Angel, "To Host a Legacy System to the Web" (1999). *Student Work*. 3595.  
<https://digitalcommons.unomaha.edu/studentwork/3595>

This Thesis is brought to you for free and open access by DigitalCommons@UNO. It has been accepted for inclusion in Student Work by an authorized administrator of DigitalCommons@UNO. For more information, please contact [unodigitalcommons@unomaha.edu](mailto:unodigitalcommons@unomaha.edu).



TO HOST A LEGACY SYSTEM TO  
THE WEB

A Thesis

Presented to the

Department of Computers Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

University of Nebraska at Omaha

by

Miguel Angel Bustamante

November 1999

UMI Number: EP74789

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

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



UMI EP74789

Published by ProQuest LLC (2015). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code

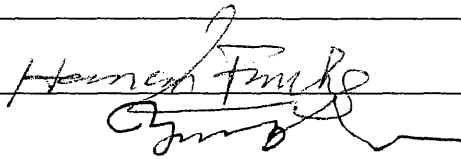
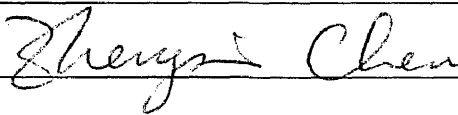


ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346

## THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College,  
University of Nebraska, in partial fulfillment of the  
requirements for the degree Master of Science,  
University of Nebraska at Omaha.

Committee

Chairperson 

Date \_\_\_\_\_

University of Nebraska at Omaha

Abstract

TO HOST A LEGACY SYSTEM TO  
THE WEB

by Miguel Angel Bustamante

Chairperson of the Supervisory Committee: Professor Stanley A. Wileman  
Department of Computer Science

The dramatic improvements in global interconnectivity due to intranets, extranets and the Internet has led to many enterprises to consider migrating legacy systems to a web based systems. While data remapping is relatively straightforward in most cases, greater challenges lie in adapting legacy application software. This research effort describes an experiment in which a legacy system International Invoice (IIMM) to a web-client/server environment. First, this thesis reports on the difficulties and issues arising when porting a legacy system International Invoice (IIMM) to a web-client/server environment. Next, this research analyzes the underlying issues, and offer cautionary guidance to future migrators and finally this research effort builds a prototype of the legacy system on a web client/server environment that demonstrates effective strategies to deal with these issues.

## ACKNOWLEDGMENTS

A year and a half ago I was taking a class with Professor Wileman. One day after a lecture I told him that I was interested in hosting an IBM mainframe system in the network and using the latest web technology. Subsequently he asked me if I have knowledge of CGI, and HTML. I said a little bit; then he told me "The first thing you should do is to research that both topics..." That was the beginning of this project. During the time of this research effort many people helped me in many aspects. My deepest thanks go out to each and every person who helped make this research possible.

**Professor Wileman**, my advisor. None of this would have happened without his guidance and support.

My Committee: **Dr. Zhengxin Chen, Dr. Hassan Farhat, Dr. Yong Shi.** Especially Professor Chen who provided me with invaluable suggestions.

**Peggy Nunamaker** (Secretary CS UNO).

My Family: **Raul, Rosa, Lorena, Michelangelo, Julia, Wilder, Francisco, Raul, Edith, Clara, and Lillian** who will always be there for me.

The **People of Valmont**: Mike Brown (Manager I/S), Bob Schulze, Mike Michalski, Jerry Toussaint, Cecil Jones, Lakshmin Ramakrishnan, Mary Parrish, Julie Zakovec, and Patricia Clausen. Particularly Mike Brown for allowed me to have a flexible schedule.

Thanks for ALL of your help.

## TABLE OF CONTENTS

<i>Chapter</i>	<i>Page</i>
TABLE OF CONTENTS.....	1
1.- Introduction .....	5
1.1. Framework.....	5
1.2. Problem Description .....	6
1.2.1 Restructuring User Interface .....	7
1.2.2 Re-Writing Database Transactions .....	7
1.2.3 Translating Language Features .....	8
1.3. Purpose of this Research.....	8
1.4. Overview of the Thesis.....	9
2.- The Legacy System .....	10
2.1. Framework.....	10
2.2. Old System General Description .....	12
2.3. The International Invoice System Hosted in the Mainframe .....	13
2.4. Conclusion .....	15
3.- Background Concepts .....	25
3.1. Introduction .....	27
3.2. Key Definitions of Internet and Intranet Technology .....	29
3.3. Concepts from Web Development, Database, and Oper. Systems .....	32
3.4. The Sun Intranet Architecture (CORBA) .....	52
3.5. The Microsoft Intranet Technology .....	56
3.6. Conclusion .....	60
4.- The New System .....	72
4.1. Introduction .....	74
4.2. Development of the prototype.....	75
4.3. Difficulties and Issues arising when creating the prototype .....	102
4.4. Cautionary guidance for future migrators .....	105
4.5. Summary.....	107
5.- Concluding comments .....	152
Reference List.....	154
Appendix A: Tables Composition .....	159

## LIST OF FIGURES

<i>Figure</i>	<i>Title</i>	<i>Pages</i>
2.1	Relationships of various user interfaces .....	17
2.2	The batch sequential style .....	18
2.3	Repository architecture for database showing control and report .....	19
2.4	Mainframe International Invoice main menu.....	20
2.5	Mainframe International Invoice header information.....	21
2.6	Mainframe International Invoice order selection screen.....	22
2.7	Mainframe International Invoice text lines.....	23
2.8	Mainframe International Invoice preview screen .....	24
3.1	Corporate Intranet .....	61
3.2	TCP/IP .....	62
3.3	An HTTP Session .....	63
3.4	Database Interfaces .....	64
3.5	Active Server Page Architecture .....	65
3.6	Server Script Use.....	66
3.7	Services Model .....	67
3.8	Service Tiers.....	68
3.9	Transaction Example .....	69
3.10	Active X Control Architecture .....	70
3.11	Java Applet Architecture .....	71
4.1	IIMM'S Home Page .....	108
4.2	Invoice and Dealer .....	109
4.3	Order Selection Screen .....	110
4.4	Selecting Lines.....	111
4.5	Header Information .....	112
4.6	Header Information (no records available) .....	113

## LIST OF FIGURES CONTINUE

<i>Figure</i>	<i>Title</i>	<i>Pages</i>
4.7	Status ready for new record .....	114
4.8	Code Terms is a list box .....	115
4.9	Employee authorization is a list box.....	116
4.10	Order removal screen .....	117
4.11	Line selection screen.....	118
4.12	Line selected.....	119
4.13	Option4_Change .....	120
4.14	Add a Detail Line .....	121
4.15	Add a Text Line .....	122
4.16	Invoice Text Lines.....	123
4.17	Prefix Lines (Lines Before Detail Lines) .....	124
4.18	Total Lines .....	125
4.19	To Change (For Valmont Industries. INC) .....	126
4.20	Maintenance Screen.....	127
4.21	Maintenance Screen (2).....	128
4.22	Preview Screen Start (A) .....	129
4.23	Preview Screen Continue (B) .....	130
4.24	Preview Screen End (C).....	131
4.25	Invoice Hard Copy First Page.....	132
4.26	Invoice Hard Copy Last Page .....	133

## LIST OF EXTRACTS

<i>Extract</i>	<i>Title</i>	<i>Pages</i>
X0	Links_htm Related to Fig. 4.1 .....	134
X1	Profile_asp Related to Fig 4.2 .....	135
X2	Page Redirection, Related to Fig 4.2.....	136
X3	Component Calling, Related to Fig 4.2.....	137
X4	Transcript_asp, Related to Fig 4.3.....	138
X5	Bid_asp, Related to Fig 4.4 .....	139
X6	ACID properties, Related to Fig 4.4.....	140
X7	Related to Fig 4.5 .....	141
X8	Related to Fig 4.7 .....	142
X9	Maria_3_asp, Related to Fig 4.10 .....	143
X10	ACID Properties, Using a Stored Procedure .....	144
X11	Bid4_asp Related to Fig 4.12.....	145
X12	Code Related to Fig 4.13.....	146
X13	Code Related to Fig 4.15.....	147
X14	Code Related to Fig 4.12 (Return, Delete, Options).....	148
X15	Rau_55htm_asp, Code Related to Fig 4.16.....	149
X16	New_Rau55b1_asp Related to Fig 4.18 (Presentation).....	150
X17	New_Rau55b1_asp Related to Fig 4.18 (Interaction) .....	151

## *Chapter 1*

### INTRODUCTION

#### 1.1 Framework

In 1508 Pope Julius II commissioned Michelangelo Buonarroti to paint the ceiling of the Cappella Sistina (Sistine Chapel) in Rome. Michelangelo labored for five years, lying on his back much of that time, to complete the task. In 1538, Pope Paul III called him back to add an enhancement, the last judgement, over the altar; this took seven more years. Michelangelo work in the Sistine Chapel is a legacy, a gift from the past that would be impossible to recreate and warrants preserving. In 1965, the Holy See commissioned a team of scientists, historians, and artists to restore the paintings. Though the restoration decision stirred controversy, work proceeded over the next 30 years. Although no software system compares to Michelangelo's masterpieces, the restoration process offers striking parallels to software reengineering. [RUGA98]

Software systems can also be legacies, though in this context "legacy" often connotes a burden rather than a gift from the past. Legacy software systems by definition have been in operation for years. Normally these applications begin life as well-designed implementations. They're useful and valuable to the organization, so over time they're "extended" in a number of ways; organizational requirements change and new logic is added. They're modified to communicate with other applications, a user interface is added, state is made persistent, log files are added, and so on. Most of these changes are infrastructure changes; they lie outside the domain of the business application and exist only to support the implementation of it. These enhancements are also typically incremental; the

new functionality is added not all at once as part of a redesign, but gradually and more or less independently of other enhancements.

Such systems usually suffer from old age in several respects: the original developers are no longer with the company, little or no documentation exists to describe the internal workings of the software, normal maintenance must continue, when the underlying hardware changes or when the operating system is upgraded, legacy software must also be adjusted, the perennial issue of bug fixes may arise. [HORO98] As a consequence, the state of the system actually drives business decisions instead of supporting them (for instance, frequently we say "We can't do that because our system won't be able to support it.").

This paper describes an approach to re-engineering and converting a system of legacy software (International Invoice IIMM) which is hosted in the mainframe to a Distribute System [new] using a Web browsers as user interface and utilizing Database Servers (ORACLE) running on microprocessors.

This research study will be taken place at **Valmont Industries, Inc.** which is a corporation located in Valley. This company leads the world in the production of mechanized agricultural irrigation equipment.

## 1.2 Problem Description

This research will have three overall constraints. First, the converted system had to be functionally equivalent to the original system. Second, there should not be any additional complexity for the user interface. Third, Since the new system will replicate functionality of the old system and we will make it as transparent a change as possible; no important user retraining will be necessary.

---

To show the complexity of the conversion problem, we first describe the crucial differences between the two technological standards.

### 1.2.1 Restructuring User Interface

The legacy user interface on a CICS teleprocessing platform is based on screen-by-screen interactions. Whenever a screen image is sent to a terminal, the program stores any data it has processed up to that point, frees the resources used, and ends temporarily. When the user enters more data and enters an attention identifier key (function keys, attention keys, or ENTER), the program is restarted along with the stored data, and processing continues until the next screen image. [MEGA95] In contrast, the new user interface in the web-based system is the client web browser software. The browser retrieves the application's home page (in HTML), which provides a high-level overview of the information resources available to the user then, after a request by the client is generated the application on the web server parses the data in the request and starts generating an HTML response when this work is finished the complete HTML response is return through HTTP to the user.

### 1.2.2 Re-Writing Database Transactions

Both on-line and batch programs in the original legacy systems used file processing for defining, searching, reading, and writing records with a local DBMS or other type of file structure. Typical file and record operations include READ, WRITE, REWRITE, START, FIND, CONNECT, RECONNECT, GET, OBTAIN, ROLLBACK, INITIATE, READY, STORE, DELETE, MODIFY, etc. In contrast, ActiveX™ Data Objects (ADO) enables us to write a client application to access and manipulate data in a database server through a provider. In ADO, the Recordset object is the main interface to data. ADO supports features for building client/server and web-based applications, including the creation of independent objects, batch updating, support for stored

---

procedures with in/out parameters and return values, different cursor types, advanced recordset cache management, support for limits on number of returned rows, support for multiple recordsets returned from stored procedures or batch statements, free-threaded objects for efficient web server applications. Moreover, during conversion we used a Session object which allows us to manage state information for one user over multiple HTTP requests.

### 1.2.3 Translating Language Features

Additionally, from the IDEAL-ADR, COBOL and ASSEMBLER environment to the Web service-based application model, there are many language "dialect" differences and environmental differences which must be converted. These changes concern all aspects of an application program. These changes are generally not difficult to understand for those well-versed in the two environments but are tedious, time-consuming and error-prone.

Most conversion instances were not one-to-one, but depended on analysis of the contextual information on their meanings and thorough knowledge of the two platforms. Much of the research in software engineering has concentrated on methods for developing new software rather than methods for analyzing, converting and evolving existing software even though studies indicate that more than half the funds expended for software in the United States are for software maintenance [LIBA98]

## 1.3 Purpose of this Research

The goals of this research are first, itemize the difficulties and issues arising when porting a legacy system to a web client/server environment. Second, after analyzing the underlying issues and difficulties offer cautionary guidance to future migrators, and finally build a prototype of the legacy system on a web

client/server environment that demonstrates effective strategies to deal with these issues.

#### **1.4 Overview of the Thesis**

The remainder of this paper is organized into five sections. In section 2 called The Legacy System, we briefly discuss about legacy systems in general and the International Invoice System in particular. In section 3 called Background Concepts we present theory dealing mainly with the web and intranet technology. Section 4 called The new system will describe the development of the prototype system hosted on a web client/server and finally Section 5 contains the conclusion and direction for future research.

## *Chapter 2*

### THE LEGACY SYSTEM

#### 2.1 Framework

Many mission-critical systems, ranging from accounting to sales tracking to air traffic control, were created years ago [MUNS98] and were typically written in COBOL. According to PC week there are over 100 billion lines of this kind of code just in the United States. For example, in the early 1980s, IBM released its DB2 relational database management system (RDBMS) for the Multiple Virtual Systems (MVS) operating system. Today, many organizations that use mainframes are still dependent on IBM's Information Management System (IMS) and Virtual Sequential Access Method (VSAM). During 1996, over 2 billion IMS transactions were still being processed daily, according to International Data Corp. The reality is that legacy systems, in one form or another, still provide most online transaction processing for many large organizations. [OLSE98]

Legacy code is code typically written in a third-generation language, such as COBOL or Fortran, [BENN95] and structured in a certain way. It is not just the language used but also the structural approach that defines legacy code. According to those practices, developers create large, monolithic systems with few separable parts. There are reportedly billions of lines of COBOL code in existence and since the majority of large-systems development were done using what are considered legacy languages, many of the existing systems in the corporate world are implemented in these languages, and most development staff members are still working with legacy code. They maintain and develop enhancements to systems typically written in COBOL, RPG, BAL, and Fortran. To perform their jobs well, they need a considerable amount of skill and practical

experience. The two main factors that support the continued use of legacy code are the huge quantity of existing code and the valuable skills of legacy developers, which have been acquired over many years of creating code.

Legacy systems represent our mainframe heritage, a significant investment of resources, and often the sticking point when trying to transition to more general 'open' Client-Server architectures. Legacy software was written years ago using outdated techniques, yet it continues to do useful work.[BENN95] Those systems often have lived for more than 15 years. The organizations owning this software by constantly adjusting the code, have created systems that meet their business demands to a high degree and since this systems went through many years of maintenance work this applications include a large amount of implicit and undocumented information and knowledge about their particular operation domain. Usually they are loaded with diverse business policies and corporate decisions and constitute one of the most important industrial assets of any company.

## 2.2 Old system general description

At Valmont Industries, Inc. we currently have a mix of a Batch Sequential Architecture and a Repository Database Architecture; hosted in an IBM system 390 (Mainframe) using MVS/ESA as our operating system.

Our users communicate with MVS, by using a simulated 3270 terminal screens, through one of several user environments that are provided as MVS subsystems or as special programs called application enablers. Communications with MVS itself are made with service requests from programs, and with Job Control Language (JCL). The most important of the user environments are TSO (the time sharing option), ISPF (the Interactive System Productivity Facility), JES (the Job Entry Subsystem), and CICS (the Customer Information Control System). Please see figure 2.1

TSO is a subsystem of MVS. It is considered part of the operating system and performs specific functions inside the operating system, as well as for the user. TSO is intended primarily for use by programmers to develop programs and to prepare jobs for execution.

JES is a subsystem of MVS. It is considered part of the operating system. It is the mechanism for the submission of jobs (one or more programs to be executed) to MVS for scheduling and execution.

ISPF is a menu-driven facility that serves as an extension to TSO. ISPF provides a series of menu screens, called panels, that offer most of the functionality of TSO in an easy-to-use form. ISPF also provides a more powerful screen editor

CICS is an application enabler. An application enabler is an application-level program that serves as a stand-in for the operating system, providing services to application programs designed specifically to work with it. As an application enabler, CICS provides the capability for the execution of on-line interactive

programs that MVS is lacking. The main function of CICS is to provide an interface between customer/client data and the programs that process data.

The part of **Batch Sequential Architecture** consists of applications in which individual database operations (transactions) are collected into large batches; then a small number of large stand-alone programs perform sequential updates on VSAM files. Usually we have first, a massive edit program, which accepts transaction inputs and performs whatever validation is possible; then a massive transaction sort, which got the transactions into the same order as the records on the sequential master file. And finally a sequence of update programs one for each master file. Please see figure 2.2

The part of **Repository Database Architecture** consists of interactive technology that allow users to do on-line processing (updates and queries). We have a repository database ( a CA-DATACOM/DB database which is similar to DB2 in that it is relational and consists of simple tables linked together by common keys that represent a column in each table) in which shared persistent data is manipulated by independent functions, each of which has essentially no permanent state. Please see figure 2.3 The on-line processing is achieved using CICS; users of CICS can transmit data from their terminals to the system where the data is processed and then sent back to the user.

### 2.3 The International Invoice System (IIMM) hosted in the mainframe

The goal of this paragraph is to orient the reader by presenting to he or she a short description of the old system (IIMM) hosted in the mainframe. The International Invoice system is a series of screens that allow the users to combine multiple orders into one International Invoice. This system produces documentation (a manifesto) of the contents of each container shipped over seas. To generate this International Invoice the users choose a subsets of multiple domestic invoices (wherever fits in the container) then they produce an

International Invoice with matches what is contained on the freight. This system is also used to browse and maintain International Invoices and to generate hard copies of specific parts of the Invoices.

The system was developed in such a way that the user is presented with menu or submenu views. The views are selection displays (screens) in which the user can make a choice to initiate a desired function for instance the figure 2.4 is an example of a menu screen (International Invoice main menu) a selection from this menu, shown in figure 2.5, takes one of the functions displayed on a menu. (selection 2 IISS) The figure 2.5 is also an example of a detail screen since it is used to display data belonging to a particular Invoice but it also facilitates the entry of data into a program during inquiry and maintenance function. At the completion of the inquiry or maintenance function, control is passed back to the control screen please see figure 2.4 Main Menu.

From the main menu the user can choose the selection number 1, International Order Selection Screen (IIOS), please see figure 2.6, this view is an example of a Browse screen since the terminal operator is allowed to view many occurrences (order numbers for a specific dealer) but it is also an example of a Selection screen since the user can select orders for a given dealer and assign an international invoice number to the orders.

From the main menu the user also can choose the selection number 5, International Invoice Text Lines (IITL), please see figure 2.7, this view is an example of a submenu screen since the terminal operator is presented with a submenu screen that allow he or she to transfer control to any of the six alternatives; after the user selects one of the alternatives that information is used to format the detail screen which is displayed next by the system.

An important option of the main menu is the alternative number 7, International Invoice Preview Screen (IITL), please see figure 2.8, this picture, which is a view

only screen, allows the operator to see the invoice that he or she is creating then by using this feedback the employee can from the main menu make the necessary changes.

This system was developed using CICS service facility; this environment permits application programs which run under CICS to issue a requests to a DATACOM/DB multiuser facility.

The application programs were mainly written using IDEAL-ADR (Applied Data Research, four generation language). IDEAL is an application development system that provides an interactive setting for the development, maintenance, and execution of complete integrated applications. Since IDEAL programs can call subprograms written in a conventional language like COBOL or ASSEMBLER some routines written in those languages were also used.

## 2.4 Conclusion

Since Legacy code contains many of the business rules describing how companies operate. This code can be translated into new languages, but that requires a very good understanding of how the legacy code functions. Such understanding may be difficult to achieve because a large percentage of these older systems may not be totally understood by the development staff, whose knowledge of the code is accomplished by trial and error and where it is typical the lack of code's documentation. However by hosting legacy applications to the web and using existing programmer knowledge with the advantages of the desktop, corporations can enjoy multiple benefits for instance, exploit new technology taking advantage of a better infrastructure that delivers solutions faster with minimal business disruption to create a competitive advantage. This approach will also allow businesses to address any Year 2000 compliance issues and to improve corporate productivity; end users and developers will be more productive because multiple platforms can now operate together seamlessly. End users can work with a

consistent graphical user interface to find information that is location independent, while developers can use the technology to implement the best solution.

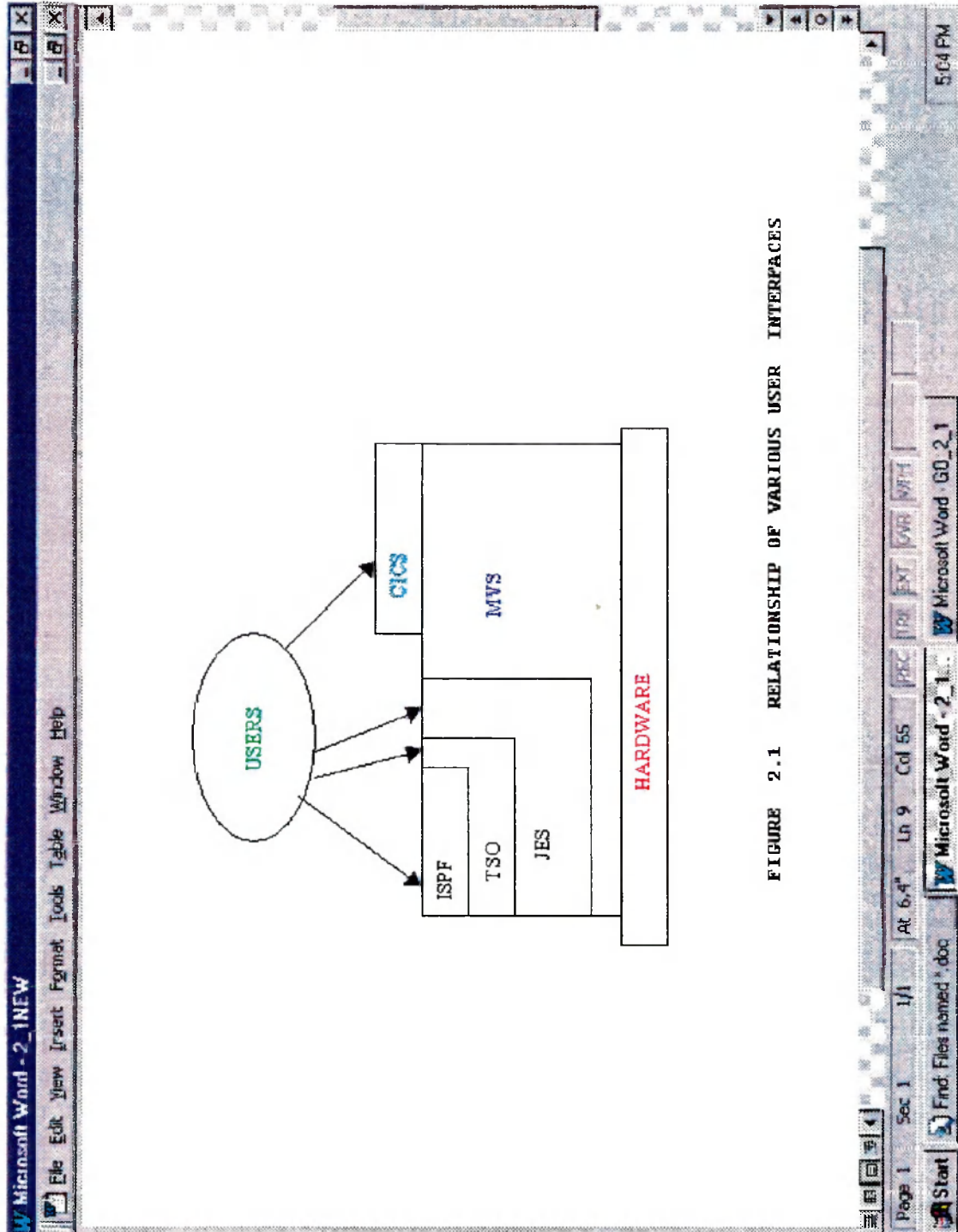


FIGURE 2.1 RELATIONSHIP OF VARIOUS USER INTERFACES

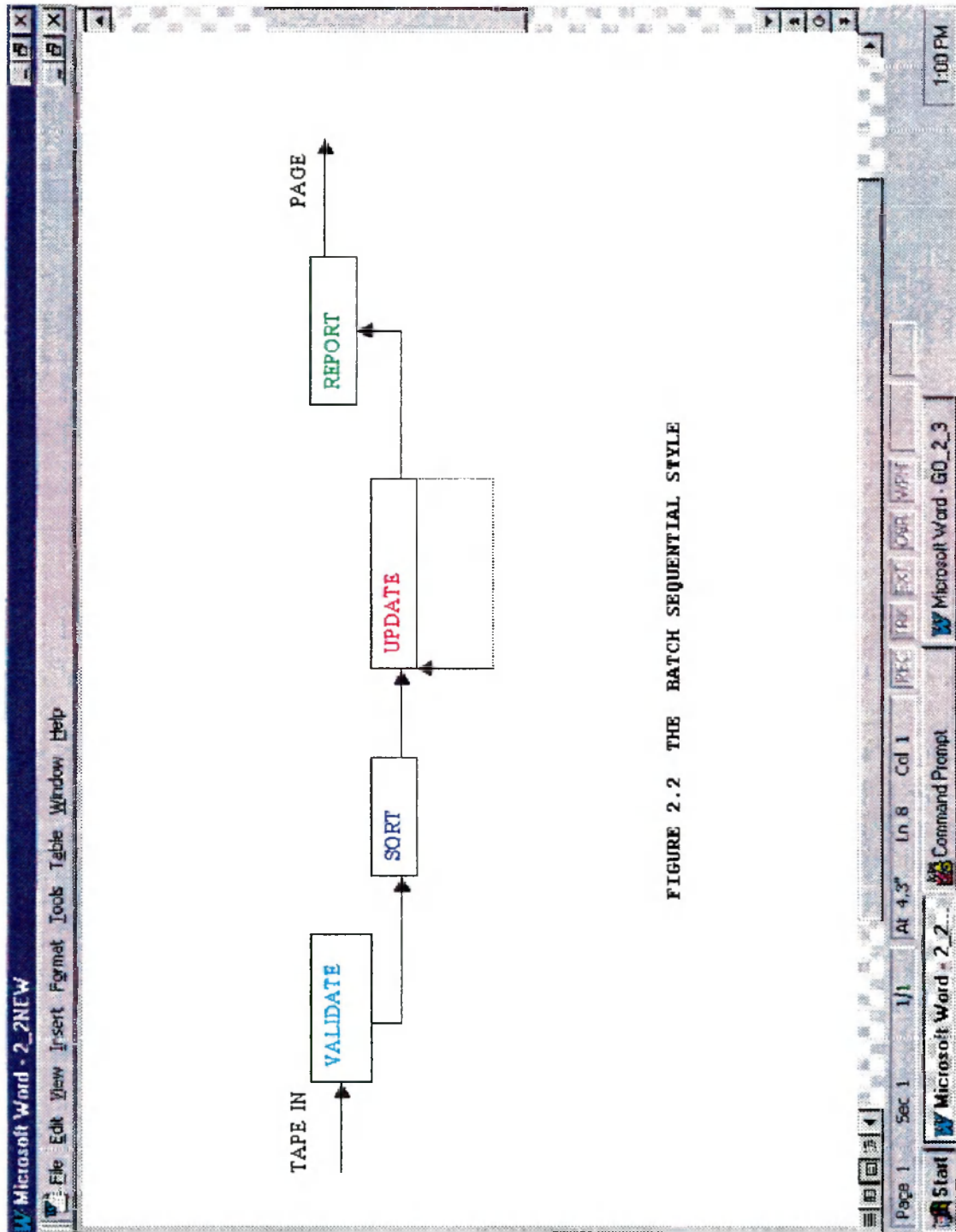


FIGURE 2.2 THE BATCH SEQUENTIAL STYLE

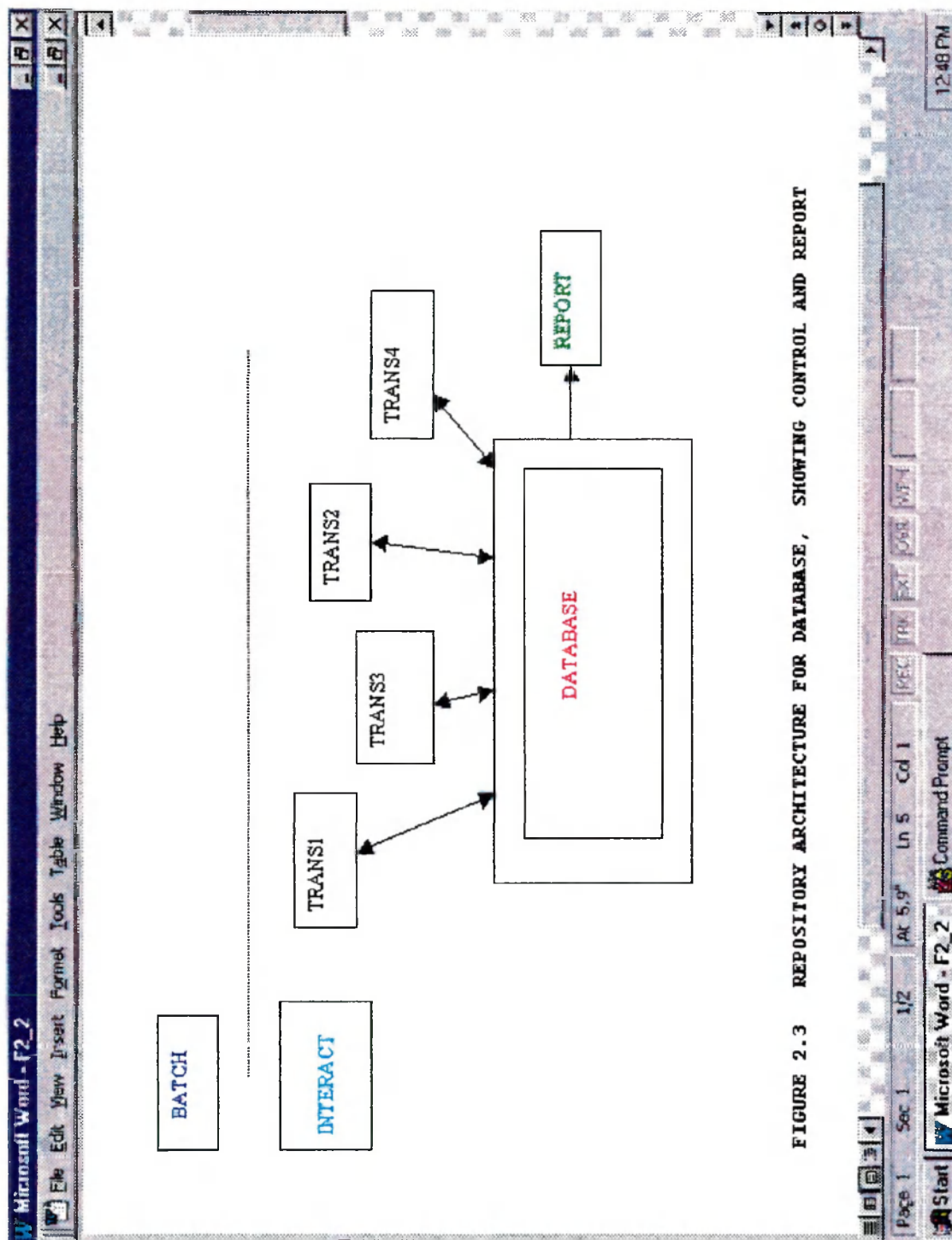


FIGURE 2.3 REPOSITORY ARCHITECTURE FOR DATABASE, SHOWING CONTROL AND REPORT

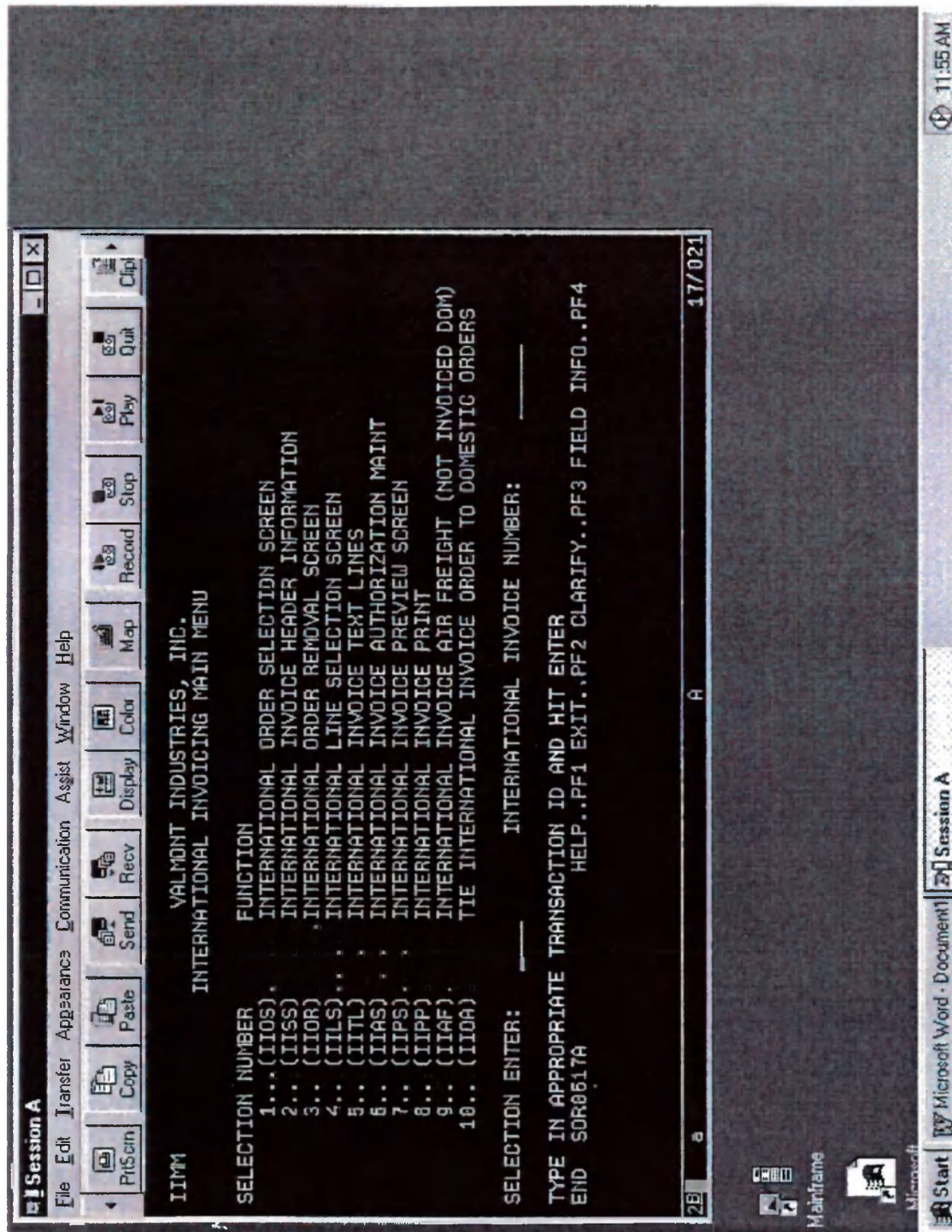


FIGURE 2.4 MAINFRAME INTERNATIONAL INVOICE MAIN MENU



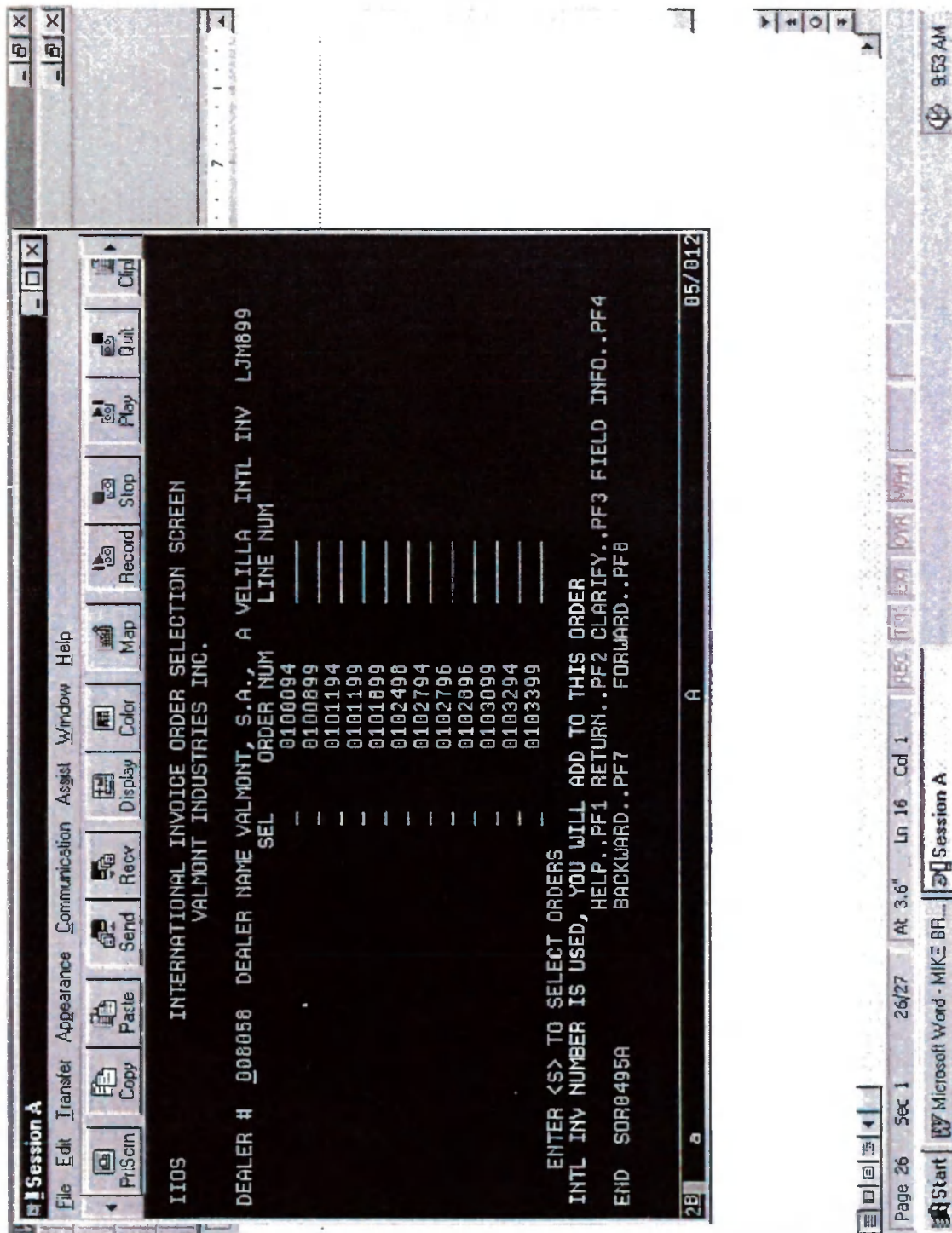


FIGURE 2.6 MAINFRAME INTERNATIONAL INVOICE ORDER SELECTION SCREEN

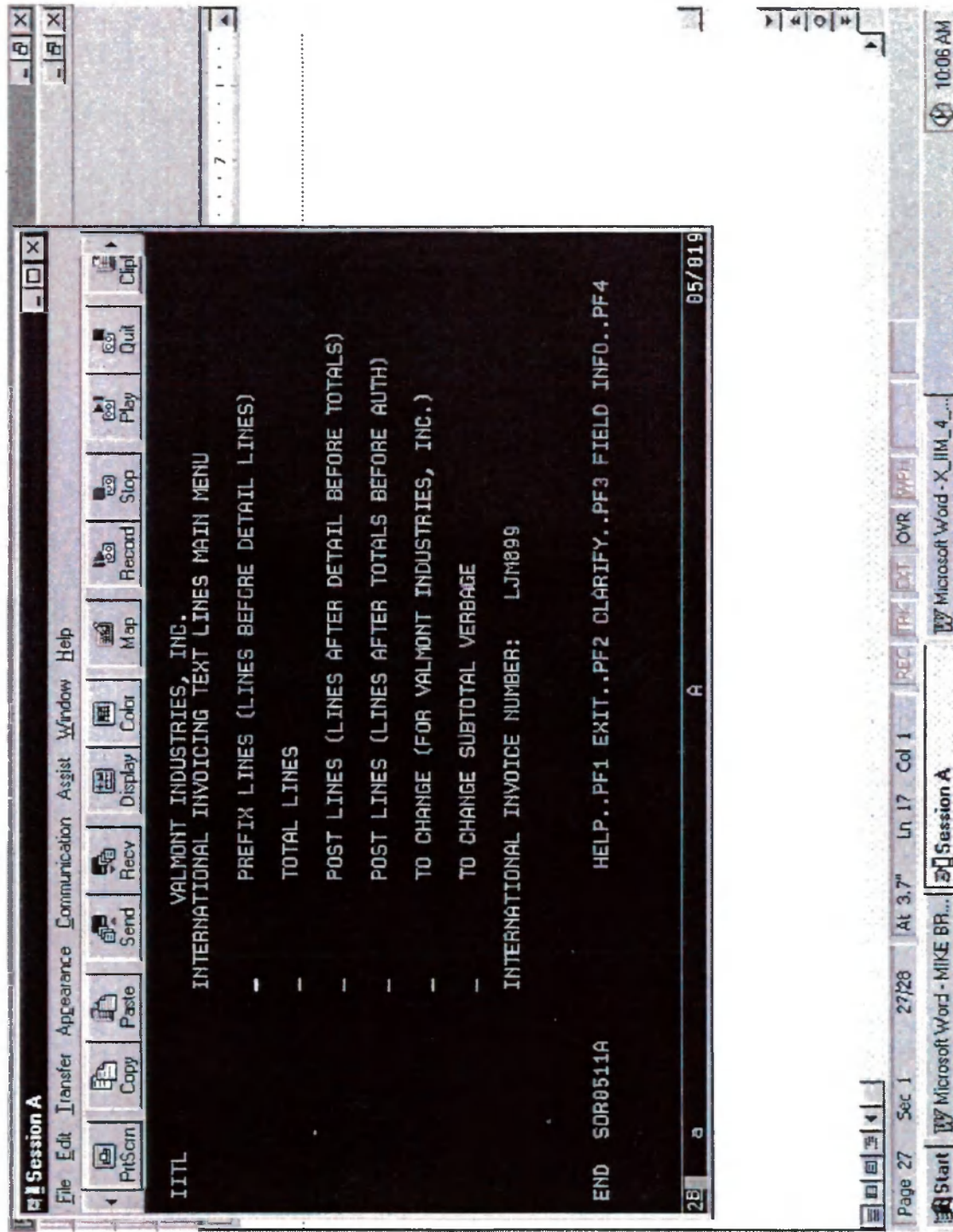


FIGURE 2.7 MAINFRAME INTERNATIONAL INVOICE TEXT LINES



FIGURE 2.8 MAINFRAME INTERNATIONAL INVOICE PREVIEW SCREEN

## Chapter 3

### BACKGROUND CONCEPTS

<i>Chapter Outline</i>	<i>Page</i>
3.1.- Introduction .....	27
3.1.1 History Of Computer Information Technology .....	27
3.1.2 The Content Era .....	28
3.1.3 Outline of the rest of the chapter .....	28
3.2.- Key definitions of Internet and Intranet technology .....	29
3.3.- Concepts from web development, database and operating systems .....	32
3.3.1 Internet, Intranet the Internet .....	32
3.3.2 TCP/IP protocol suite .....	33
3.3.3 An HTTP Session .....	36
3.3.4 Database Technologies .....	36
3.3.4.1 Types of Database Interfaces .....	36
3.3.4.1.1 Open Database Connectivity (ODBC) .....	36
3.3.4.1.2 OLE DB .....	36
3.3.4.2 Technologies to access a Database .....	36
3.3.4.2.1 ActiveX Data Objects (ADO) .....	37
3.3.5 Client Vs Server Scripting .....	38
3.3.5.1 Client-Side Script .....	38
3.3.5.2 Server-Side Script .....	38
3.3.5.3 Common Tasks of Client and Server Script .....	39
3.3.5.4 Browsers And Scripting Languages .....	39
3.3.5.4.1 VBScript .....	40
3.3.5.4.2 JScript and JavaScript .....	40
3.3.6 Introduction To Active Server Pages (ASP) .....	41
3.3.6.1 Difference Between ASP And HTML Pages .....	41
3.3.6.2 Active Server Page Architecture .....	41
3.3.6.3 Main Tasks Performed With Asp .....	41
3.3.6.4 Server Script Use .....	42
3.3.7 Services Model .....	43
3.3.7.1 User services .....	43
3.3.7.2 Business services .....	43
3.3.7.3 Data services .....	43
3.3.7.4 Benefits Of Using A Service-Based Application .....	43
3.3.7.4.1 Better manageability .....	44
3.3.7.4.2 Isolation of functionality .....	44

3.3.7.5	Basic Concepts of Business Services .....	44
3.3.7.5.1	Business Rules .....	44
3.3.7.5.2	Business Processes.....	44
3.3.8	Service Tiers .....	46
3.3.8.1	Single-Tier Applications .....	46
3.3.8.2	Two-Tier Client/Server Applications .....	46
3.3.8.3	Three-Tier Client/Server Applications .....	47
3.3.9	Transaction Processing Concepts .....	48
3.3.9.1	The ACID Test .....	48
3.3.9.1.1	Atomicity .....	48
3.3.9.1.2	Consistency .....	48
3.3.9.1.3	Isolation .....	49
3.3.9.1.4	Durability .....	49
3.3.9.2	Commit or Abort Transactions .....	49
3.3.10	Active X Control Architecture .....	50
3.3.10.1	ActiveX Controls on a Web Page .....	50
3.3.11	Java Applet Architecture .....	51
3.3.11.1	Java Applets on a Web Page .....	51
3.4.-	The Sun intranet architecture (CORBA) .....	52
3.4.1	Introduction .....	52
3.4.2	The Common Object Request Broker Architecture (CORBA).....	52
3.4.3	Sun's Approach To The Creation Of Intranet .....	53
3.4.3.1	Java .....	53
3.4.3.2	JavaBeans .....	54
3.4.3.3	Java Database Connectivity (JDBC) .....	54
3.4.3.4	Java Remote Method Invocation (RMI) .....	54
3.4.3.5	Java Interface Definition Language (Java IDL) .....	55
3.4.4	Summary .....	55
3.5.-	The Microsoft intranet architecture .....	56
3.5.1	Introduction .....	56
3.5.2	Microsoft ActiveX.....	56
3.5.2.1	ActiveX On The Server .....	56
3.5.2.2	ActiveX on the Server (Composition) .....	57
3.5.3	The most common ActiveX technologies.....	57
3.5.3.1	ActiveX Automation .....	57
3.5.3.2	ActiveX Control .....	57
3.5.3.3	ActiveX Component .....	57
3.5.3.4	ActiveX Server Component .....	58
3.5.3.5	ActiveX Scripting .....	58
3.5.4	A Three Tier Approach Built Using Microsoft Technology .....	58
3.5.5	Summary .....	59
3.6.-	Conclusion.....	60

### 3.1 Introduction<sup>1</sup>

To place what we have each experienced in the past few years in perspective consider that our ancestors used the first primitive implements of information technology to record their experiences on the walls of their caves. It was thousands of years before the Egyptians gave us a formal means of written communication (3000 B.C.) and not until 105 A.D. that the Chinese invented paper in roughly the form that we know it today. Guttenberg invented the printing press 1,300 years later, and Edison invented the telephone four hundred and fifty years after that (in 1876). All remarkable contributions to communications technology, but perhaps none more remarkable than we have experienced in the past 50 years with the development of computer information technology.

#### 3.1.1 History Of Computer Information Technology

The history of computer information technology can be loosely classified into three major phases, or eras:

- a.- The Hardware Era, which began in 1945 with the development of the Electronic Numerical Integrator and Calculator (ENIAC).
- b.- The Software Era, which began in 1975 with the introduction of the Altair 8800, the first "personal" computer for the general public.
- c.- The Content Era, which represents one of the most significant changes brought by the Internet, accentuates the creation and management of content rather than application software logic.

---

<sup>1</sup> Source: Rick Tanler. (1997) The Intranet Data Warehouse. Wiley Computer Publishing (First Edition) extracted from the Preface

### 3.1.2 The Content Era

The transition into the Content Era, began with the popularization of the Internet and World Wide Web. In the Content Era, the focus shifts to the content rather than the hardware and software technologies that are used to deliver it. Unlike the Software Era in which users were tasked with loading applications from diskette (or CD-ROM) and then using the application logic to access or create useful content, the Internet (and the World Wide Web) presents users with content first, hiding the application software from view. Users receive only the application logic that they need to support the content and retain only the content and application logic that they require at their local workstation.

### 3.1.3 Outline of the Chapter

Since the main body of this research paper deals with the porting of a legacy system to the Web. In this chapter we first present some key definitions of Internet and Intranet technology, next we present, abbreviated, some concepts belonging to the areas of web development, database, operating systems and programming languages. (Later in chapter 4 this paper will use those concepts) Next, we briefly present the two main technologies that developers are using to create a distributed network-based solution: a.- The Sun intranet architecture which supports the Common Object Request Broker Architecture (CORBA). b.- The Microsoft intranet architecture which advocates a distributed network computing strategy based on its ActiveX platform and Distributed Component Object Model (COM, DCOM) and finally we conclude this section with some comments about the many similarities between CORBA and COM approaches for the creation of a distributed network-based solution.

### 3.2 Key definitions of Internet and Intranet technology

Active Server Pages (ASP)	It is a server-side scripting environment that we can use to create and run dynamic, interactive Web server applications.
ActiveX	It is a set of integration technologies. It helps applications and components communicate and coordinate with one another, whether on one machine, across a LAN, or over the Internet. ActiveX technologies include ActiveX Controls, ActiveX Automation, and ActiveX Scripting.
ActiveX Data Objects (ADO)	It enables us to write a client application to access and manipulate data in a database server through a provider.
API (Application Programming Interface)	A set of routines that an application program uses to request and carry out lower-level services performed by a computer's operating system.
CGI (Common Gateway Interface)	The first widely supported standard for linking external programs to HTTP; it provides the link between HTTP and HTML to external programs that dynamically generate documents
COM (The Component Object Model)	It is the low-level foundation on which ActiveX (and OLE) is built. It is an object-based programming model designed to allow components written by different vendors to interoperate in a well known and consistent manner. It is a binary standard, meaning that it can operate across platforms and can be implemented using any program language.
DCOM (The Distributed Component Object Model)	It is COM extended to work over a network. DCOM allows components to communicate directly with one another, regardless of their physical location. Using DCOM, applications can be implemented as a collection of components that reside on different machines connected through a LAN, WAN, or even the Internet.
DLL (Dynamic Link Libraries)	It allows executable routines, generally serving a specific function or set of functions, to be stored separately as files with DLL extensions and to be loaded only when needed by the program that calls them.

DNS (Domain Name System software)	A system of hierarchical names and name servers that translates host names into Internet addresses.
HTML (HyperText Markup Language)	The language commonly used to create text pages for Web display. HTML incorporates codes that define fonts, page layout, embedded graphics, and hypertext links. These codes, which are generally called tags instruct the browser on how to display content.
HTTP (HyperText Transfer Protocol)	One of the most common methods of transferring documents across the Internet. The syntax of a typical HTTP URL is <u>http://host/path</u> . For example, the URL <u>http://www.microsoft.com</u> defines the browser protocol as HTTP: the host hierarchy is a named server "microsoft" within the World Wide Web domain. Because HTTP uses a standard port number (80), no port number is specified.
Hypertext links	The pointers that link HTML pages for display on Web browsers. Hypertext links include the destination URLs. The Web browser highlights text that has a hypertext pointer and interprets a click on the text as a request to access the referenced document.
Internet Information Server (IIS).	It is Microsoft's Internet server product; it provides standard Web, FTP, and Gopher services.
Internet, Intranet	Internet is the collection of networks and gateways that use the TCP/IP suite of protocols. Intranet is the Internet but for a particular organization.
IP (Internet Protocol)	A low-level communications protocol that creates the basic links between nodes. Most Web servers rely on IP.
ODBC and DAO	Both Data Access Objects (DAO) and Open Database Connectivity (ODBC) are application programming interfaces (APIs) that give us the ability to write applications that are independent of any particular database management system (DBMS).
OLAP (Online Analytic	It provides the online data analysis capabilities to answer decision makers' ongoing flood of questions. E. F. Codd

Processing)	introduced the concept of Online Analytic Processing in a White paper published in 1993.
OLE (Object Linking and Embedding)	It has a long history, has been effectively replaced by the term ActiveX. At its beginning, OLE was a technology that enabled documents to be embedded within and linked to one another.
TCP (Transmission Control Protocol)	A higher-level communications protocol that operates in conjunction with IP to provide a reliable virtual connection between two applications situated anywhere on the network. Together the IP and TCP protocols are commonly referred to as TCP/IP.
URL (Universal Resource Locator)	Essentially, the address of an Internet resource, along with information about how it can be accessed. The URL uses a schemepath syntax in which scheme identifies the protocol that the browser uses to access the resource (e.g., HTTP), and path is a hierarchical name that includes the host name and an optional port number.
Web browser	A program that issues requests for resources across networks and displays the resources when they are returned.

### 3.3 Concepts from web development, database and operating systems

The goal of this section is to present concepts with the help of examples and graphics. This concepts will tie the isolated definitions that we presented in section 3.2 And this topics will help to better understand the development of the prototype in chapter 4.

#### 3.3.1 Internet, Intranet the Internet<sup>2</sup>

A network is a collection of computers connected by a data link medium, such as Ethernet or Token Ring. An internet is a network of possibly incompatible networks, enabling connected users to share resources. An intranet is simply one organization's internet. Please see figure 3.1. The term the Internet refers to the worldwide research-oriented network that includes more than 10,000 networks and makes use of several different search tools.

While each network has its own method of communicating among its host computers, there must be a standardized way for all networks on an internet to communicate with each other. The TCP/IP protocol suite is the standard the Internet uses.

The purpose of any network or internet is to transfer semantic data between host computers. The data is meaningful only to the applications residing on the hosts; the network itself sees it only as data. The TCP/IP protocols encapsulate the host computer's data and add header information to allow routing of the data.

A program interaction with a network can be conceptualized as the activation of a series of layers, starting at the application layer and ending at the network's physical layer, where the data is sent as a stream of bits "across the wire."

---

<sup>2</sup> Source: Microsoft Mastering C++ Chapter 3

### 3.3.2 TCP/IP protocol suite<sup>3</sup>

The TCP/IP protocol suite is a set of protocols used to send data through the Internet. TCP/IP usually is the underlying transport layer used for socket communication. Other protocols, such as FTP and HTTP, also use the TCP protocol as its underlying transport layer.

The TCP/IP suite is hierarchically structured. Please see figure 3.2. Protocols that provide sophisticated services depend on simpler protocols lower in the hierarchy. For instance, the Transmission Control Protocol (TCP) relies on a simpler network layer protocol, the Internet Protocol (IP) layer, which only attempts to deliver data to a destination computer. IP is an unreliable protocol. There is no guarantee provided at the IP layer.

It is the more sophisticated TCP layer that provides a reliable communication connection between computers, guaranteeing that data will be successfully transferred. TCP guarantees the transfer of data by using a strategy employing handshaking, timeouts, retries, and checksum calculations.

---

<sup>3</sup> Source: Microsoft Mastering C++ Chapter 3

### 3.3.3 An HTTP Session<sup>4</sup>

Hypertext Transfer Protocol, or HTTP, is an Internet protocol supporting distributive, collaborative, and hypermedia information systems: distributive, because one hypertext document may have links to many servers; collaborative because the client and server work together, through the protocol, to present the best possible presentation of the data to the user; and hypermedia, because the protocol is independent of the content being transferred.

The protocol sits atop the TCP/IP layer, or some other reliable transport layer, and is based on the request/response paradigm. The figure 3.3 shows an HTTP session and the process that occurs when a user clicks a hyperlink within a hypertext document, the following sequence of events occurs.

- i. The embedded hyperlink string is parsed for the server name, the path on the server, and the file name.
- ii. The client establishes a connection to a server. (The browser creates a TCP/IP connection to the server)
- iii. The client sends a request, usually asking for a file. (The browser packages a request for an HTML document from the server into an HTTP request message, and then sends the message to the server by using a TCP/IP connection)
- iv. The server responds with the data of that file. (The server receives the HTTP request and processes it based on the request method contained in the request line. The server then sends back an HTTP response message. Part of the response message is a status line that contains code indicating whether the attempt to satisfy the HTTP request was successful)

---

<sup>4</sup> Source: Microsoft Mastering Web application development chapter 4

- v. The server closes the connection. (When the Web browser receives the HTTP response message, the TCP/IP connection is closed, and the HTTP session terminates)

### 3.3.4 Database Technologies<sup>5</sup>

Database systems generally provide a driver that conforms to a standard interface. This enables a user to access the database by using any technology that also supports the interface.

#### 3.3.4.1 Types of Database Interfaces

The most common standard database interfaces are ODBC and OLE DB. Please, see figure 3.4

##### 3.3.4.1.1 Open Database Connectivity (ODBC)

ODBC is the standard interface to relational database systems. Most database systems provide an ODBC driver, which enables a user to access a database from any tool that works with ODBC-compliant databases. This interface is used specifically for SQL databases.

##### 3.3.4.1.2 OLE DB

OLE DB is a set of interfaces for universal data integration, regardless of the data type. For example, OLE DB can be used to access both SQL databases as well as any other type of database that provides an OLE DB driver.

#### 3.3.4.2 Technologies to access a Database

There are several technologies that a user may utilize to access a database from a Web page for instance the Internet Database Connector (IDC), the Advanced Data Connector (ADC) and the ActiveX Data Objects. (ADO) From this technologies ADO provides the most flexibility since ADO runs on the Web server and returns straight HTML text. ADO can work with any browser.

---

<sup>5</sup> Source: Microsoft Mastering Web application development chapter 6

#### 3.3.4.2.1 ActiveX Data Objects (ADO)

ADO provides the most flexibility of the database access technologies. ADO is a collection of Automation objects that can retrieve, update, and create records in any OLE DB or ODBC database. Please, see figure 3.4 The user can utilize ADO from any Automation controller, such as .asp files and ActiveX server components.

Because ADO uses Automation, The user can utilize ADO in any application or program that is an Automation controller, such as Active Server Pages, Microsoft Internet Explorer, Visual Basic, and Visual C++.

ADO provides a layer between the Web page and the underlying database. To work with a database, The user writes code that sets properties and invokes methods of ADO objects.

ADO communicates with a database driver by using OLE DB, which is a COM-based technology. OLE DB can access both SQL-based and non-SQL data. If a database provides an OLE DB provider, ADO communicates directly with the provider. If the database provides an ODBC driver, ADO communicates through MSDASQL.DLL to the driver.

### 3.3.5 Client Vs Server Scripting<sup>6</sup>

One of the first scripting languages was JCL (used to sequence Job Steps). The individual Job Steps were written in PL/1, Fortran, or Assembler languages. [OUST98]

Scripting is code that a person writes in a Web page. The script runs either on the client computer when a user interacts with a control, or on the Web server before the page is returned to the client. In both cases, a person adds script to a Web page as ASCII text.

#### 3.3.5.1 Client-Side Script

Client-side script runs on the client computer. Web browsers contain scripting interpreters that can read and run the script. The primary purpose of adding client-side script to a Web page is to create event procedures for controls. For example, a person can write an event procedure that runs a function when a user clicks a particular button.

Client-side script is not compiled, nor is it encrypted on an HTML page. Therefore, if users view the HTML source of a Web page, they will see the script included in the page.

Client-side script requires a Web browser to support the scripting language. If a Web browser does not support the scripting language, the user will not have full access to the features on the Web page.

#### 3.3.5.2 Server-Side Script

Server-side script runs in an Active Server Page on a Web server before the server

---

<sup>6</sup> Source: Microsoft Mastering Web site development chapter 6

returns the page to the user. When a user requests an Active Server Page, the server-side script runs and generates HTML to return to the user. Server-side script is not visible to the user on the returned Web page.

Because server-side script runs on a Web server, the script has access to all resources that reside on that server, such as databases and executable files.

Server-side script requires a Web server to support Active Server Pages. Active Server Pages is a Microsoft Internet Server Application Program Interface (ISAPI) technology. A Web browser does not need to provide any additional functionality. Therefore, server-side script will run regardless of which Web browser is used.

#### **3.3.5.3 Common Tasks of Client and Server Script**

When a person uses client-side script, the processing occurs on the user's computer while the user interacts with the Web page. Client-side script is useful for performing the following tasks: Validating control values on a form, and providing event procedures for controls.

When a person uses server-side script, processing occurs on the Web server before the page is returned to the user. Server-side script is useful for performing the following tasks: Accessing a database and returning data to the user, and saving status information about a user or session.

#### **3.3.5.4 Browsers And Scripting Languages**

Most major Web browsers currently support one or more of the three scripting languages: VBScript, JScript, and JavaScript. Web browsers must include a scripting interpreter for the particular language. Internet Explorer 4.0 and later has interpreters for VBScript, JScript, and JavaScript. Netscape Navigator provides an interpreter for JavaScript.

#### **3.3.5.4.1 VBScript**

VBScript is a subset of Microsoft Visual Basic for Applications.

#### **3.3.5.4.2 JScript and JavaScript**

JScript is the Microsoft implementation of the ECMAScript standard. It is fully compliant with the standard, and based on JavaScript. JavaScript is a C-like language that was created by Netscape Communications Corporation.

### 3.3.6 Introduction To Active Server Pages (ASP)<sup>7</sup>

Active Server Pages provides a server-side scripting environment for the creation of interactive Web applications. An Active Server Page is a file with the extension .asp. This file can contain any combination of the following elements: Text, HTML tags, and server-side script. The .asp extension tells the Web server that the page contains server script that it should process before returning the page to the browser.

#### 3.3.6.1 Difference Between ASP And HTML Pages

The main difference between ASP and HTML pages is the location where the script is run. DHTML, or client script, is run on the client, in the browser, after the page is sent from the server. ASP, or server script, is run on the server before the page is sent to the browser. The Web server processes the script and generates the HTML pages that are returned to the Web browser.

#### 3.3.6.2 Active Server Page Architecture

An Active Server Page script runs when a browser requests an .asp file from a Web server. The Web server calls Active Server Pages, which runs scripts and commands, and then sends an HTML page to the browser.

The figure 3.5 shows how Active Server Pages are processed between the Web browser and the Web server. Static HTML pages display the same information each time a user views the page in a browser, while Active Server Pages are dynamic and interactive. Whenever a user requests an .asp file, the script runs and returns an HTML response to the user.

#### 3.3.6.3 Main Tasks Performed With Asp

---

<sup>7</sup> Source: Microsoft Mastering Web site development chapter 2

ASP provides a server-side scripting environment that enables us to perform the following tasks: a.- Read information from an HTTP request. b.- Customize an HTTP response. c.- Store information about a user. d.- Extract the capabilities of the user's browser.

#### **3.3.6.4 Server Script Use**

Server-side script is contained in either `<% %>` delimiters, or in a `<SCRIPT>` tag with the RUNAT attribute set to Server. Please see figure 3.6. When a Web server processes Active Server Pages, it runs any script code between the `<%` and `%>` delimiters.

In general, an Active Server Page contains HTML code mixed with server-side script. The server-side script programmatically determines what information will be returned to the user.

For instance in the figure 3.5 the script uses the Now and Hour procedures contained in the `<%` and `%>` delimiters determine the current time, and then greet the user with either "Good Morning" or "Good Day," depending on the time.

---

### **3.3.7 Services Model<sup>8</sup>**

When designing a Web site, it can be used a service-based application model. The term service-based means that the functionality of an application is specified as collections of services that meet specific user needs. A service-based application is typically comprised of three categories: user services, business services, and data services. Please see figure 3.7

#### **3.3.7.1 User services**

User services provide an application with its user interface. The user of a service can be a person or another service. Therefore, the interface for a service can provide a graphical user interface or a programmatic interface.

#### **3.3.7.2 Business services**

Business services enforce business rules and handle transactions. These services may impose constraints or apply transformations to change user input or raw database information into usable business information.

#### **3.3.7.3 Data services**

Data services provide storage and low-level manipulation of data in a database. Examples of data services include create, read, update, and delete, which are used by business services to modify a database. A business service does not need to know where data is located, how it is implemented, or how it is accessed. These tasks are handled by data services.

#### **3.3.7.4 Benefits Of Using A Service-Based Application**

---

<sup>8</sup> Source: Microsoft Mastering Web application development chapter 2

The main benefits of Using A Service-Based Application are Better manageability and Isolation of functionality

#### **3.3.7.4.1 Better manageability**

Because services divide the functionality of your Web site into distinct tasks, any changes in the implementation of one service will not introduce changes to another service component.

#### **3.3.7.4.2 Isolation of functionality**

The functionality of a specific service is encapsulated, so any error in the implementation of a service can be easily traced to the corresponding component.

#### **3.3.7.5 Basic Concepts of Business Services**

The two basic concepts of business services are business rules and business processes.

##### **3.3.7.5.1 Business Rules**

A business rule is a piece of heuristic logic that provides guidelines for how a piece of information should be processed. For example, if a credit card number is entered into a form, one business rule may check the credit card number for the limit available in that account. If the available credit limit exceeds the transaction, credit will be granted and the transaction will proceed.

##### **3.3.7.5.2 Business Processes**

A business process is a sequence of related tasks that produce a specific response to a user's request. For example, when a user submits an order form to purchase a product from an online catalog, a transaction is executed. This is a business

process. Most of the time a business process acts on a business rule.

### 3.3.8 Service Tiers<sup>9</sup>

Tiers are a logical concept that provide a way to describe how applications can be segmented into services, specifically the three types of services discussed in The Services Model — user, business, and data.

The three types of tiers are generally described as user (first), business (second or middle), and data (third) service tiers. The concept of tiers emphasizes the logical segmentation of the services, and is not about implementing the services nor about the number of physical computers involved in deploying the solution. The figure 3.8 shows an example of single-tier, two-tier, and three-tier client/server application.

#### 3.3.8.1 Single-Tier Applications

A single-tier application is simply a monolithic, stand-alone program that runs on the user's computer. It may communicate with a database, but that database resides on the same computer (or perhaps on a mapped network drive). The key point about a single-tier application is that all three services — user, business, and data — are architecturally combined into a single program.

#### 3.3.8.2 Two-Tier Client/Server Applications

In this type of application, the database (and perhaps a portion of the data services) is separated from the user interface and business logic. Typically, the database is placed on a dedicated server. Two-tier client/server applications are the most common type of client/server applications built today. They offer significant benefits over single-tier applications because data processing is centralized and becomes a shared resource among potentially many users.

---

<sup>9</sup> Source: Microsoft Mastering Web site development chapter 7

### 3.3.8.3 Three-Tier Client/Server Applications

Three-tier client server applications put another layer between the users and the database — the application server. This type of central application service can manage network traffic and database server loads more efficiently. Typically, the application layer handles most of the business services, and may be implemented on its own server computer, separate from the database. One of the main advantages of a three-tier architecture is the ability to extract the business logic from the user and data tiers and into the middle tier, where it is easier to maintain.

### 3.3.9 Transaction Processing Concepts<sup>10</sup>

From a business point of view, a transaction is an action that changes the state of an enterprise. For example, a customer depositing money in a checking account constitutes a banking transaction.

The figure 3.9 presents an illustration of a transaction, in this picture three business objects work together to transfer money from one account to another. The Debit object debits an account and the Credit object credits an account. The Transfer object calls the Debit and Credit objects to transfer the money.

#### 3.3.9.1 The ACID Test

A transaction changes a set of data from one state to another. For a transaction to succeed, it must have the following properties, commonly known as the ACID (Atomicity, Consistency, Isolation, and Durability) test.

##### 3.3.9.1.1 Atomicity

Atomicity means that a transaction is an indivisible unit of work: all of its actions succeed or they all fail.

##### 3.3.9.1.2 Consistency

Consistency means that after a transaction executes, it must leave the system in a correct state or it must abort. If the transaction cannot achieve a stable end state, it must return the system to its initial state.

In the transaction described earlier, money can be debited from one account and not yet credited to the other account during the transfer process. When the

---

<sup>10</sup> Source: Microsoft Mastering Web application development chapter 8

transaction is finished and able to commit, either both the debit and credit occur, or neither occurs.

#### **3.3.9.1.3 Isolation**

Isolation ensures that concurrent transactions are not aware of each other's partial and uncommitted results. Otherwise, they might create inconsistencies in the application state.

#### **3.3.9.1.4 Durability**

Durability ensures that committed updates to managed resources (such as database records) survive communication, process, and server system failures. Transactional logging enables you to recover the durable state after failures.

### **3.3.9.2 Commit or Abort Transactions**

Commit or Abort Transactions provide an all-or-nothing simple model for managing work. Either all of the objects succeed and all of the work is committed, or one or more of the objects fail and all of the work is aborted. For example, in the transaction described earlier in this topic, both databases must be changed successfully or the entire transaction will fail and the objects will be rolled back to their previous states. In either scenario, any database tables or files affected by the work will either all be changed, or not changed at all. They will not be left in an inconsistent state.

### 3.3.10 Active X Control Architecture<sup>11</sup>

ActiveX controls (formerly known as OLE controls) are component objects that can be inserted on Web pages or other ActiveX container applications. These controls can be built with multiple languages, such as C, Visual C++, Visual Basic, and Java. ActiveX controls are typically built as dynamic link libraries (DLL), and are compiled with an ocx extension. Please see figure 3.10

#### 3.3.10.1 ActiveX Controls on a Web Page

To run an ActiveX control on a Web page the control must already be installed and registered on the user's computer. If the control requires additional functionality from DLLs, the DLLs must also be installed on the user's computer. To insert controls on a Web page you can use the HTML `<OBJECT>` tag and set the CODEBASE attribute.

When Microsoft Internet Explorer encounters an `<OBJECT>` tag in HTML code, it searches for the control on the user's computer. If the control has not been installed, Microsoft Internet Explorer downloads the control from the location specified with the CODEBASE attribute, and then installs the control.

---

<sup>11</sup> Source: Microsoft Mastering Web application development chapter 4

### 3.3.11 Java Applet Architecture<sup>12</sup>

The Java language, a derivative of C++, was developed by Sun Microsystems, Inc. to be a robust, secure, and cross-platform language for use on the Internet. Java can be used to create stand-alone applications and applets.

Applets are a type of object that you can use to create an interactive HTML page. Java applets are small, reusable programs that expose an interface and are run in a certain type of container, such as a Web browser. Please see figure 3.11

#### 3.3.11.1 Java Applets on a Web Page

When you compile Java source (.java) files, the Java compiler creates Java bytecode (.class) files. When these .class files are downloaded to the user's computer, they are interpreted locally by the Java Virtual Machine (JVM).

To add Java applets on an HTML page, you use the `<APPLET>` tag. The `CODE` parameter of this tag specifies the .class file of the applet. To run an applet in a Web browser, the user's computer must have the JVM installed, and the browser's security must be set to enable the applets to run.

---

<sup>12</sup> Source: Microsoft Mastering Web application development chapter 4

### **3.4 The Sun intranet architecture (CORBA)**

#### **3.4.1 Introduction**

The Sun intranet architecture which supports the Common Object Request Broker Architecture (CORBA) a specification authored and published by The Object Management Group (OMG) which is a consortium of companies (including IBM, Digital Equipment Corporation, Hewlett-Packard, and Sun Microsystems) that have allied together to develop an object communication standard; [KOBO98] all support the CORBA specification and have released Object Request Broker (ORB) products.

#### **3.4.2 The Common Object Request Broker Architecture (CORBA)**

CORBA is a competing standard for distributed object computing. It defines an abstract object model that describes components and their interfaces. It also provides standard mappings from the abstract object definition to concrete programming languages.

CORBA provides standardized mechanisms for objects to communicate with each other. They specify how components and applications can be built in a distributed and organized fashion. CORBA provides for communication between objects distributed across a network, providing for complex multi-tiered applications based on the standard.

Perhaps most importantly, CORBA encourages the use of business objects in their architectures. And it advocates a separation of business rules from the presentation layer, in such a way that the main application logic is contained in an object model that's largely independent of the user-interface.

### 3.4.3 Sun's Approach To The Creation Of Intranet

Sun's answer to the creation of intranet involves both software and hardware solutions. Sun's intranet data warehousing strategy focuses on three critical aspects:

- i.- Scalable SMP Server hardware platforms.
- ii.- Superior network communications.
- iii.- Java as a peer-to-peer, highly distributed, applications development tool.

Sun created Java, a high level programming language. Sun also introduced JavaBeans, a component object model for building Java components and dynamically assembling them into applications. Sun also presented a number of communication capabilities including Remote Method Invocation (RMI), and Java database connectivity (JDBC).

Since the understanding of Sun's network computing model is directly related to the understanding of Java, JavaBeans, JDBC, Java RMI and Java IDL interaction this paper will present a brief description of these topics and how they relate to each other.

#### 3.4.3.1 Java

Java is a object-oriented programming (OOP) language used in Web development to provide system services not available through HTML. When we program with Java, we can create small applications, or applets, that run as plug-ins for our Web site. Java is an object-oriented programming (OOP) language that compiles into what is referred to as byte code. Resident on a machine that runs a Java applet is an application the Java virtual machine (VM) that verifies and interprets these byte codes. applets run in a protected space known as the

sandbox. The sandbox prevents the applet from accessing files and selected system services, thus protecting the system from viruses that might be resident in an applet.

#### **3.4.3.2 JavaBeans**

JavaBeans provides the component architecture that lets programmers create reusable objects (beans) that can be used without internal modification in Java programs. For any component architecture, there are three characteristics that make it useful to the end programmer: methods, properties, and events. A method is any function in a component. A property is a value that is set in a component. Events include mouse clicks, property changes, and keystrokes. They allow an application that contains a component to receive notification when things happen to that component. JavaBeans has quickly gained market attention, emerging as the dominant competitor to DCOM. Whereas DCOM is language-neutral but platform-dependent, JavaBeans is platform-neutral but language-dependent. [KRAD98]

#### **3.4.3.3 Java Database Connectivity (JDBC)**

Java Database Connectivity (JDBC) is an ANSI SQL-2 entry-level compliant database access interface that provides Java programmers with an interface, which is uniform, to a wide range of relational databases.

#### **3.4.3.4 Java Remote Method Invocation (RMI)**

Java Remote Method Invocation (RMI) is a standard for communicating between components when both the client side and the server side components are written in Java. RMI allows developers to write Java objects that are invoked from Java code running in other virtual machines, including those running on other computers systems.

#### 3.4.3.5 Java Interface Definition Language(Java IDL)

Java Interface Definition Language(Java IDL) it provides a way to transparently connect Java clients to network servers using the OMG standard Interface Definition Language. The Java IDL system permits developers to define remote interfaces in the interface definition language, then compile the IDL definitions to generate Java interface definitions and Java client and server stubs.

#### 3.4.4 Summary

In conclusion the Sun intranet architecture provides a rich hardware and software development environment for building and deploying network computing solutions and its most compelling feature is the Java's platform independence. Java allows a single application to run on multiple platforms. Further the JavaBeans component API, which is written entirely in Java, allows Java components to be inserted in any other component architecture.

Note: The above comments about the sun intranet architecture are just a short introduction. For a deeper review of this technology please visit <http://java.sun.com>.

### **3.5 The Microsoft intranet architecture**

#### **3.5.1 Introduction**

The Microsoft intranet architecture advocates a distributed network computing strategy based on its ActiveX platform and Distributed Component Object Model (DCOM). In The following paragraphs this paper will define and expose the relationship among these concepts, and at the end this writing will present a three tier approach for a solution built using the Microsoft technology.

#### **3.5.2 Microsoft ActiveX**

Microsoft ActiveX is a set of technologies that allow developers to build business applications for the Internet and intranets that run across multiple platforms the core technology elements of ActiveX are COM and DCOM. Since the terms OLE, COM, and DCOM are often mentioned in the same context as ActiveX it will be relevant to clarify the concept of each of these terms Object Linking and Embedding (OLE) is a set of integration standards to transfer and share information among client applications. Component Object Model (COM) is the object-oriented programming model that defines how objects interact within a single application or between applications. Distributed Component Object Model (DCOM) is the Component Object Model (COM) plus additions to facilitate the transparent distribution of objects over networks and over the Internet. DCOM is independent of language and implementation. DCOM was restricted primarily to the windows platform; however maturing ports to UNIX are reducing DCOM'S platform dependency. [KRAD98]

##### **3.5.2.1 ActiveX On The Server**

ActiveX on the server is the server-side platform that provides clients with Internet services, applications, data, and content. Applications and content can be hosted on the same server that runs the Web Server or on other systems. Data

can be hosted in any database that supports Open Database Connectivity (ODBC). With ActiveX on the server, developers can write Common Gateway Interface (CGI) applications, Internet Server application programming interface (ISAPI) Dynamic Link Libraries (DLLs) that run on the server side, and scripts.

### **3.5.2.2 ActiveX on the Server (Composition)**

ActiveX on the Server consists of: Microsoft Internet Information Server, which supports CGI applications, ISAPI DLLs, and connectivity to databases. Developers can write applications using the programming languages C, C++, and Java and run these applications across different platforms. Developers can take full advantage of the features provided by the targeted operating system with ActiveX controls written in many different languages. When a user clicks on a Web page that references an ActiveX control, that control is downloaded from the server and runs on the end-user's system.

### **3.5.3 The most common ActiveX technologies**

#### **3.5.3.1 ActiveX Automation**

It is language-neutral way to manipulate an ActiveX Component's methods from outside an application. ActiveX Automation is typically used to create components that expose methods to programming tools and macro languages.

#### **3.5.3.2 ActiveX Control**

It is a compiled software component based on the component object model (COM) that encapsulates a set of business or user interface functions. An ActiveX Control is used to provide user interface functionality and is designed to run on the client computer.

#### **3.5.3.3 ActiveX Component**

It is a compiled software component based on the component object model (COM) that encapsulates a set of business functionality. The functionality in an ActiveX component is accessed through ActiveX Automation interfaces. The ActiveX Component can execute either on a client computer or on a server computer, transparent to the calling application, through DCOM.

#### **3.5.3.4 ActiveX Server Component.**

It is an ActiveX Component designed to run on the server-side of a client/server application.

#### **3.5.3.5 ActiveX Scripting.**

It is the act of using a scripting language to drive ActiveX Components.

### **3.5.4 A Three Tier Approach Built Using Microsoft Technology.**

A three tier approach for a solution built using Microsoft technology will use a service based application model. First, the user services will provide the application with its interface. Second, the Business Services enforce business rules and handle transactions. Third, the Data Services provide low-level manipulation of data in a database. Finally, an application with this type of architecture functions in the following manner:

- i.- The user views an HTML page in Internet Explorer. The page may have embedded controls to prompt for the data, and may use client-side script code to gather the input and pass it along to the server.
- ii.- The user inputs are packaged into a string of parameters, concatenated onto the end of a URL requesting an Active Server Page, and sent to the server via HTTP.

iii.- An Active Server Page receives those parameters then instantiates COM objects to perform complex analysis. These objects may reside on the Web Server ( in which case they are communicated via COM) or on another server (in which case they are communicated via DCOM).

iv.- The COM objects communicate with one or more databases via Microsoft's Open Database Connectivity (ODBC) protocol and perform whatever logic is pertinent to their function.

v.- The resulting data is dynamically published as HTML documents and sent back to the browser for viewing. The data may also be viewed in embedded controls and further manipulated at the client through additional script code.

### **3.5.5 Summary.**

The Microsoft Intranet Architecture provides a robust set of tools to implement solutions over the Internet or the corporate intranet and the key ingredients of the Microsoft architecture are: a browser client hosting HTML documents; the business logic implemented as a COM object on the server, data stored in a Server databases and ASP scripts tying everything together.

Note: The above comments about the Microsoft Intranet Architecture are just a short introduction. For a deeper review of this technology please visit <http://www.microsoft.com>

### 3.6 Conclusion

The Object Management Group (OMG) is a consortium of companies that have banded together to develop an object communication standard. The Common Object Request Broker Architecture (CORBA) is a standard that competes with Microsoft's Component Object Model (COM) in many ways.

Although it is true that there are differences between CORBA and COM, It is also authentic that there are many similarities between these two approaches for instance both CORBA and COM provide standardized mechanisms for objects to communicate with each other. They specify how components and applications can be built in a distributed and organized fashion. CORBA, like DCOM, provides for communication between objects distributed across a network, providing for complex multi-tiered applications based on the standard.

Perhaps most importantly, CORBA and COM both encourage the use of business objects in their architectures. And they both advocate a separation of business rules from the presentation layer, in such a way that the main application logic is contained in an object model that's largely independent of the user-interface.

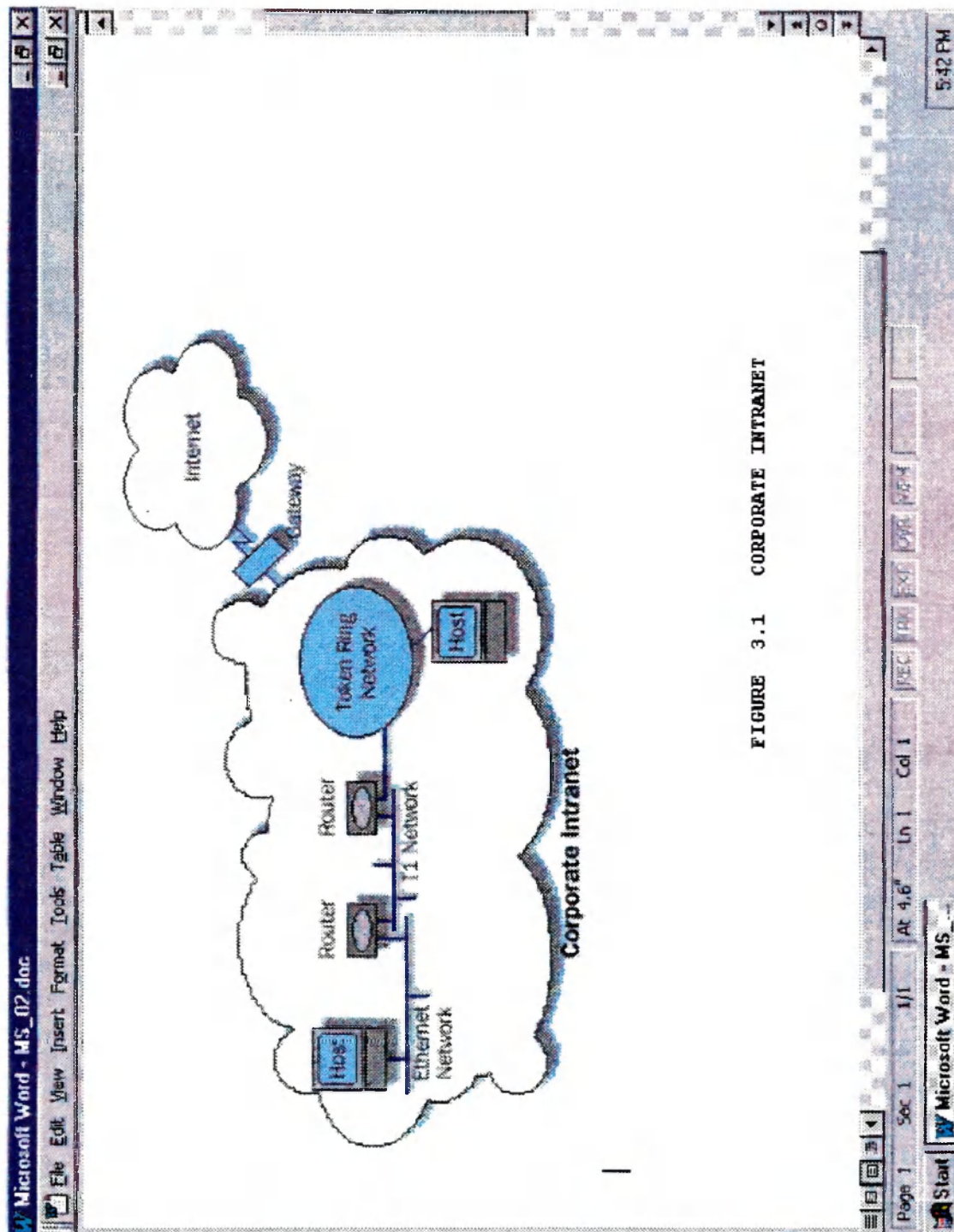


FIGURE 3.1 CORPORATE INTRANET

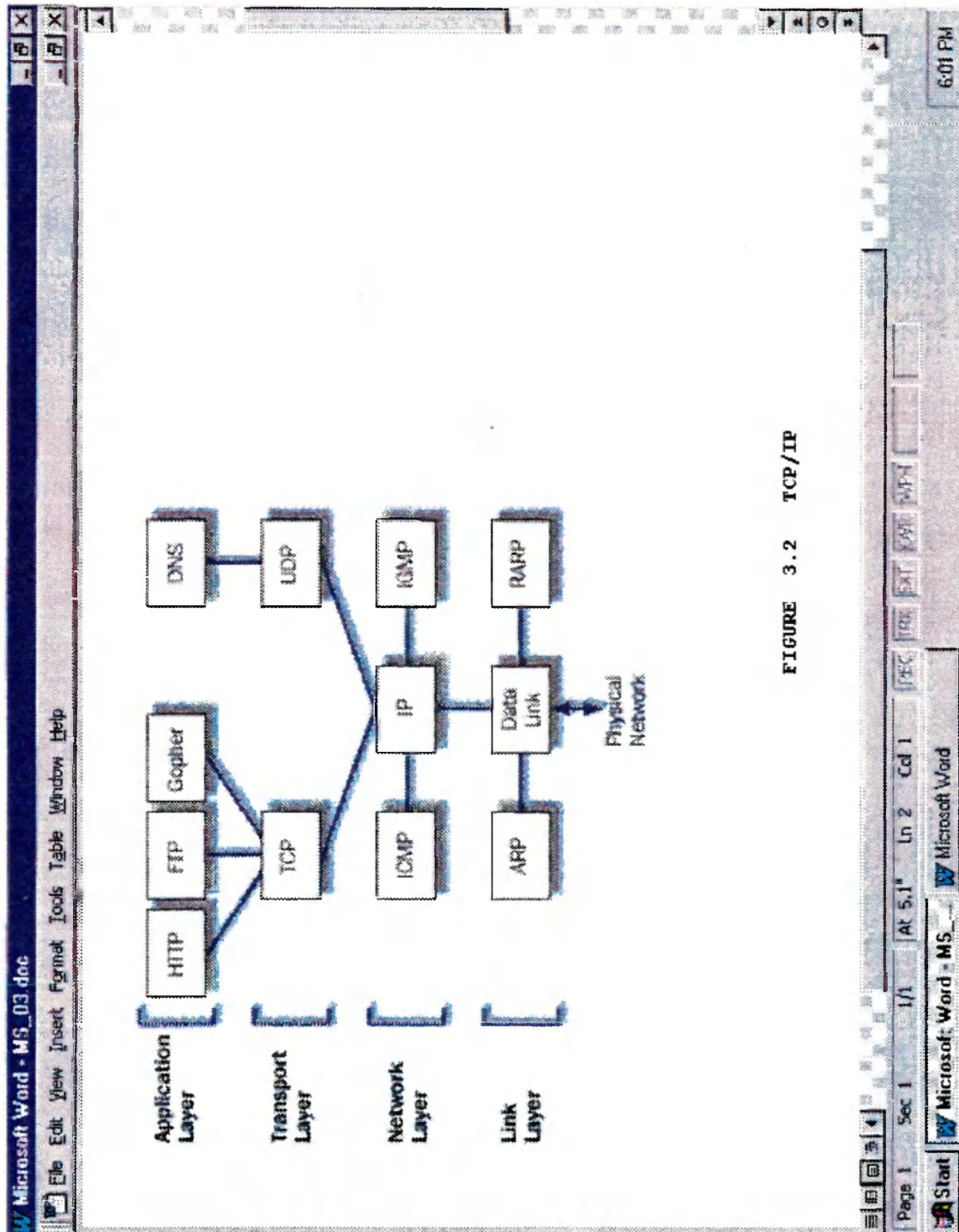


FIGURE 3.2 TCP/IP

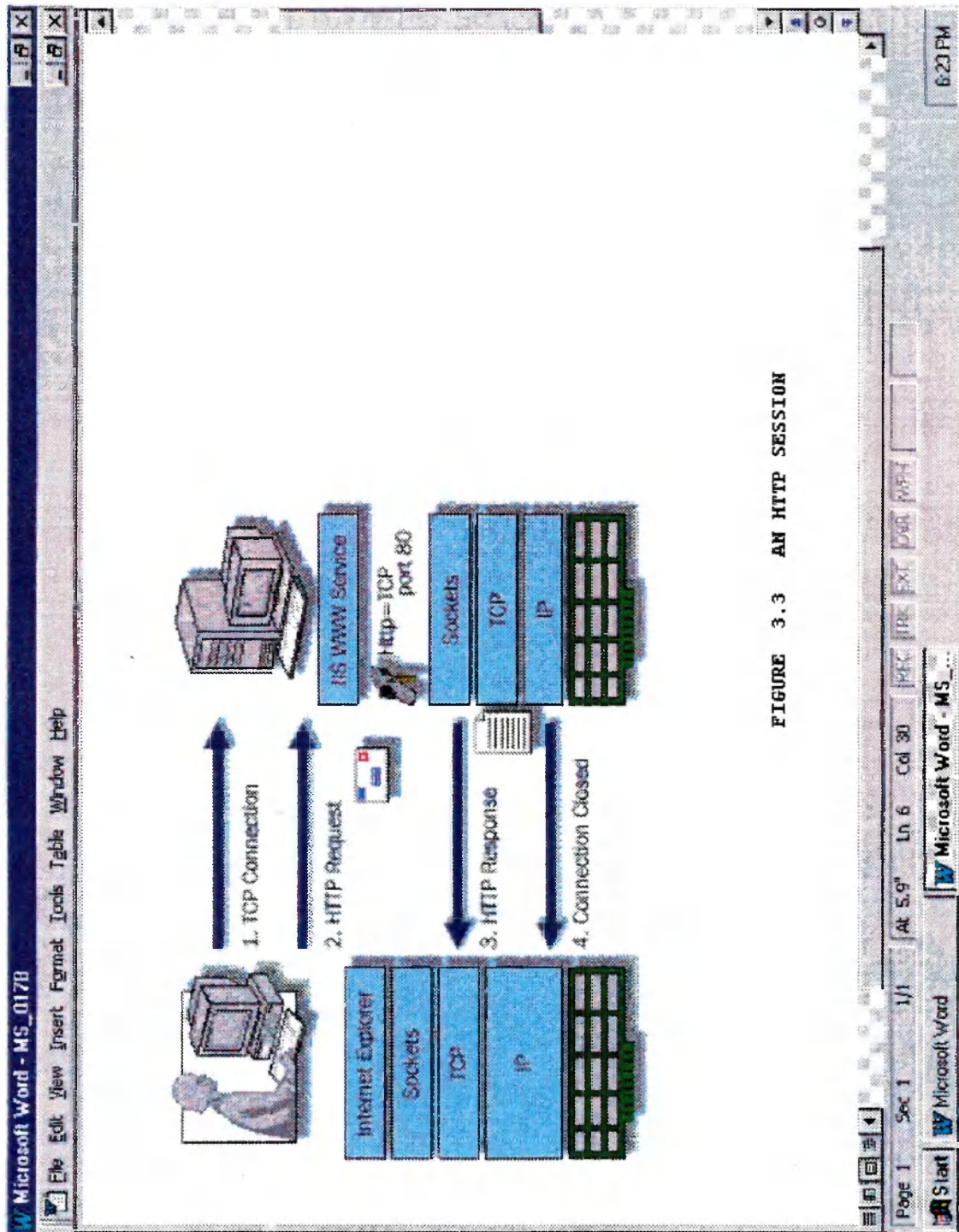


FIGURE 3.3 AN HTTP SESSION

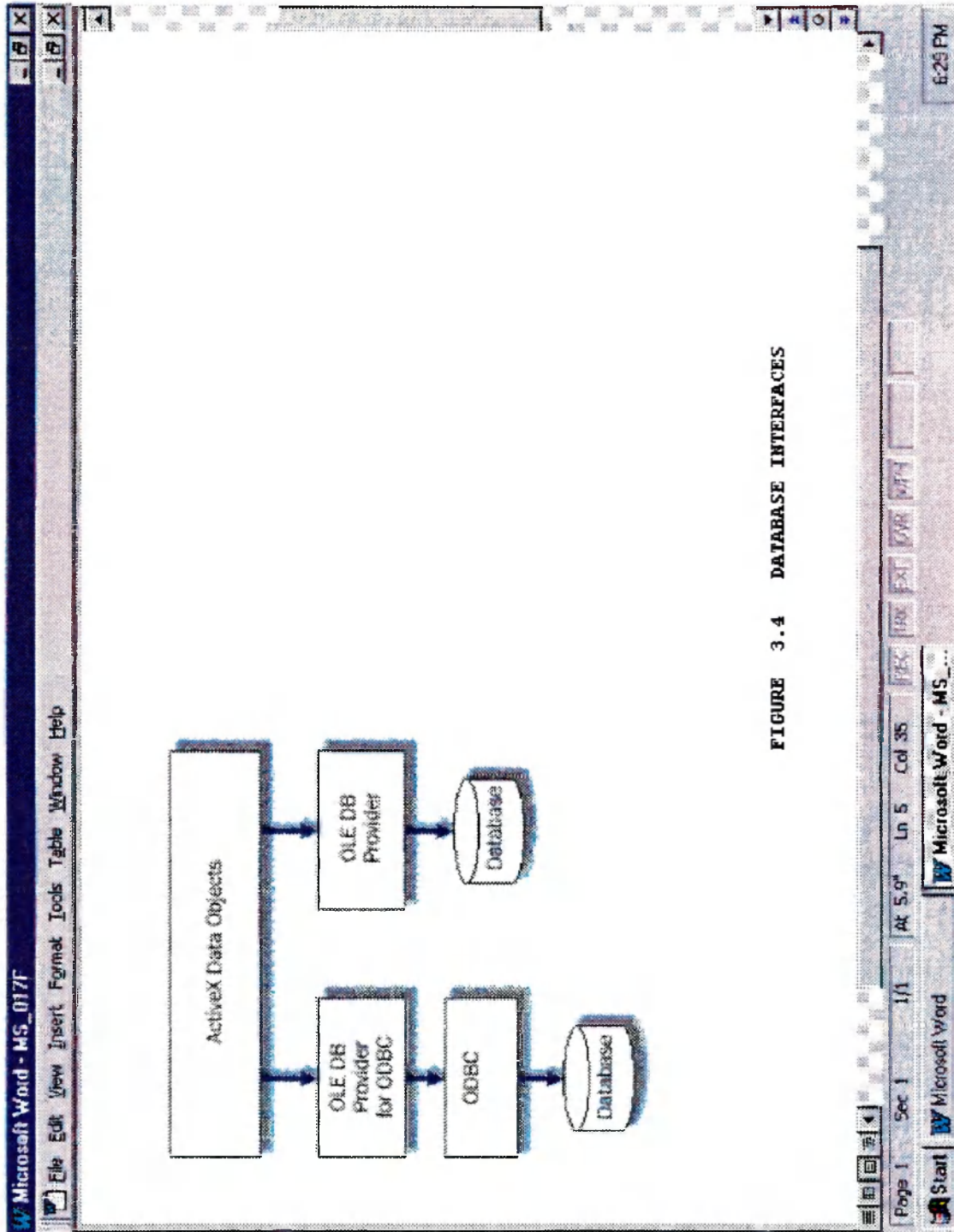


FIGURE 3.4 DATABASE INTERFACES

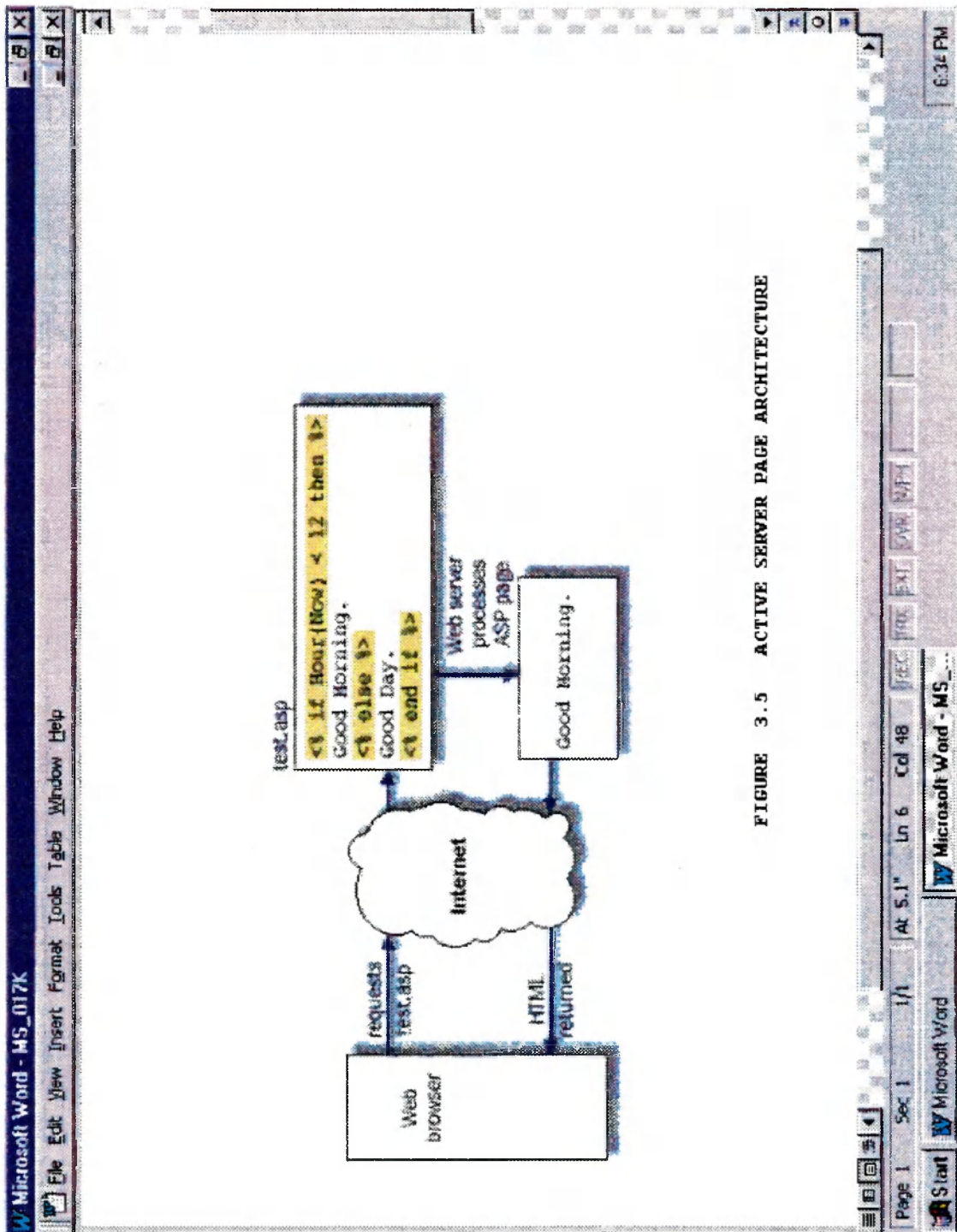


FIGURE 3.5 ACTIVE SERVER PAGE ARCHITECTURE

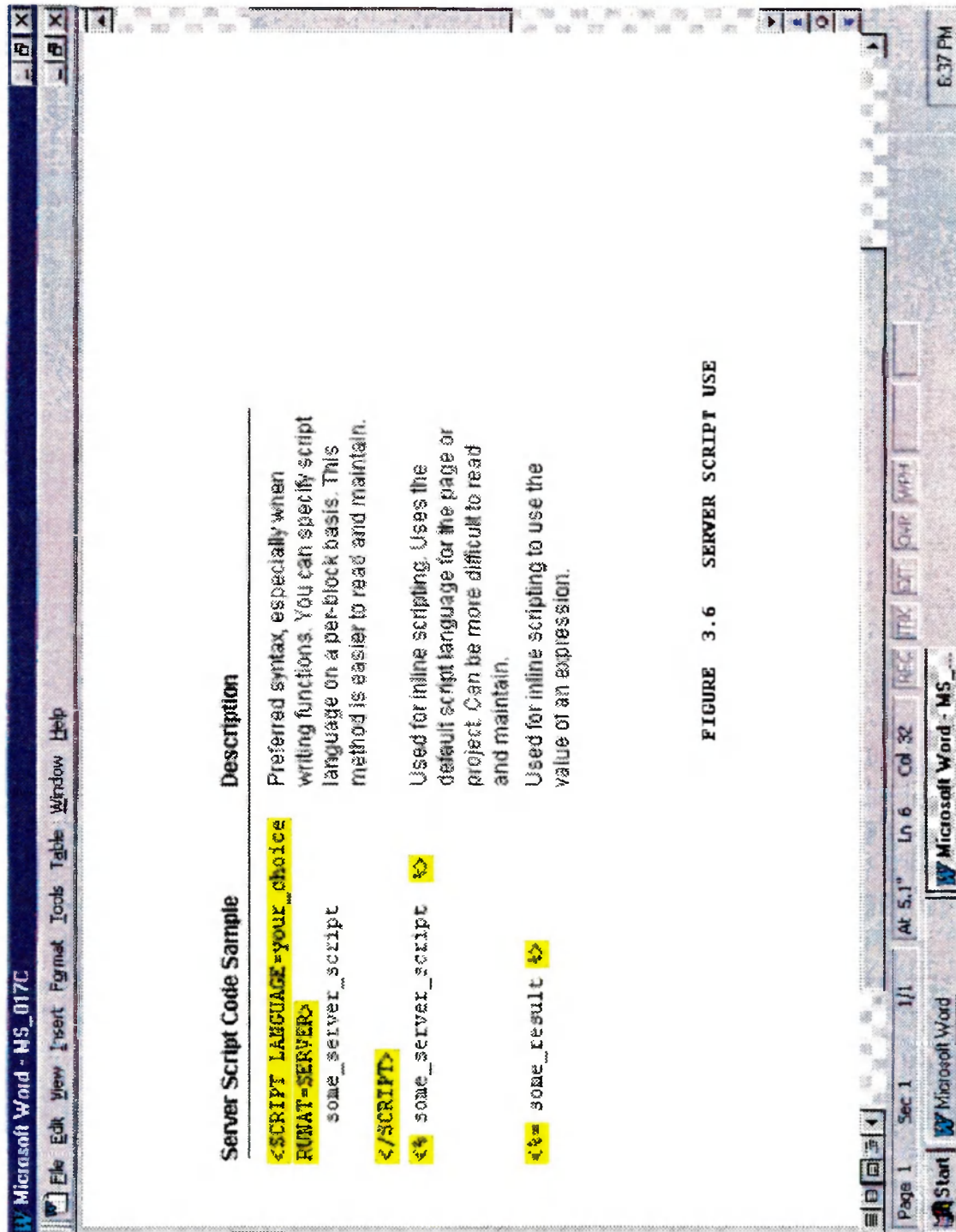


FIGURE 3.6 SERVER SCRIPT USE

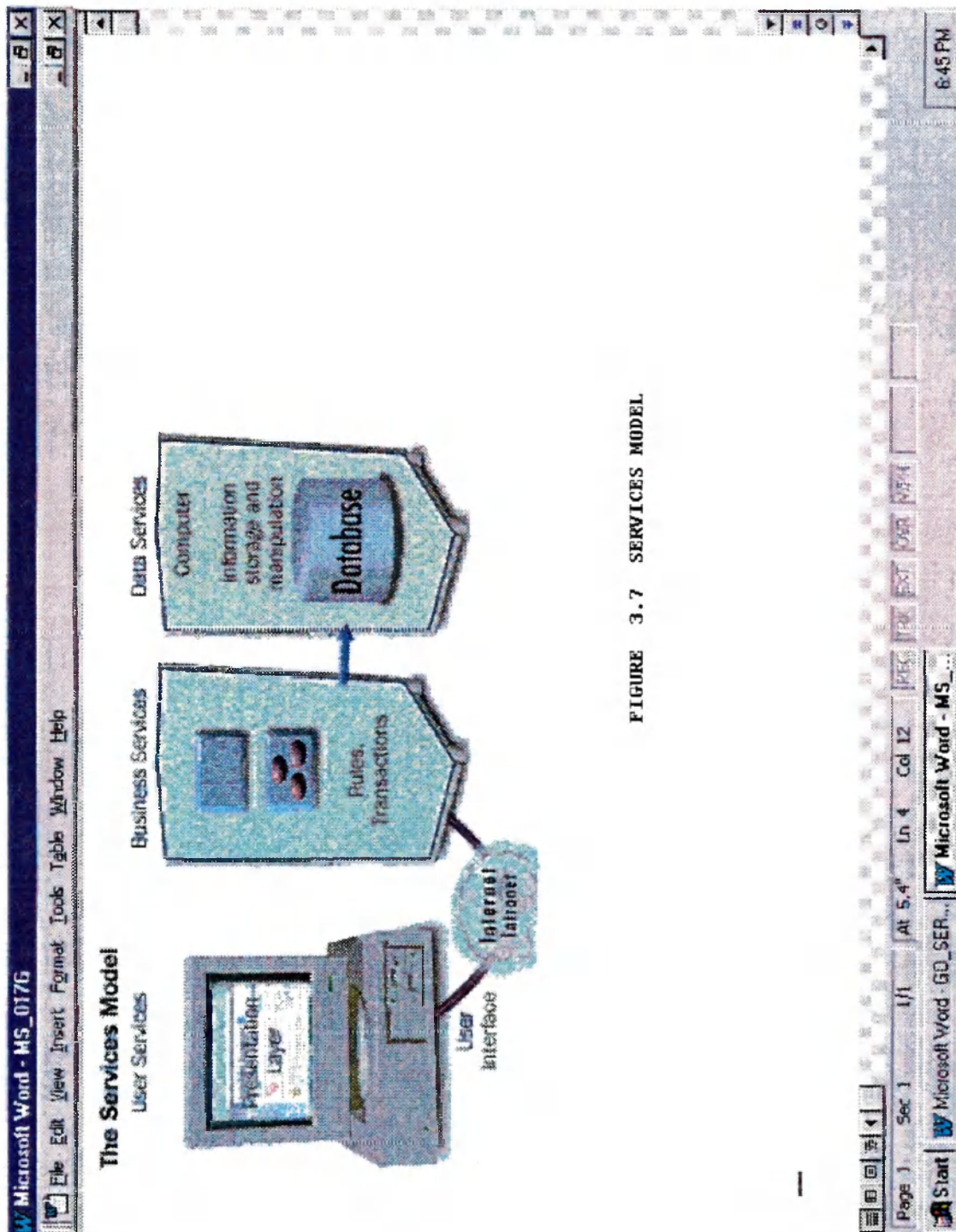


FIGURE 3.7 SERVICES MODEL

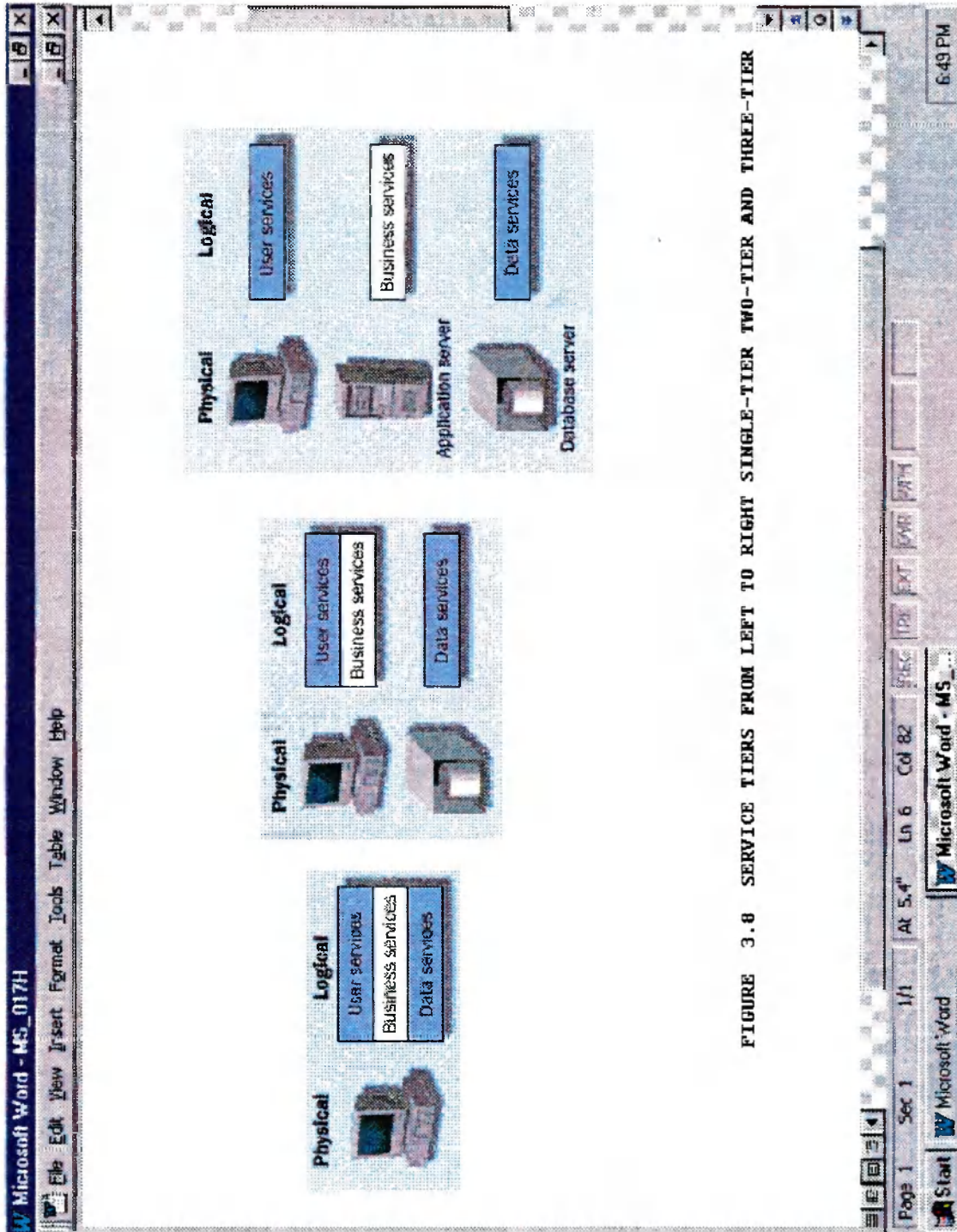


FIGURE 3.8 SERVICE TIERS FROM LEFT TO RIGHT SINGLE-TIER TWO-TIER AND THREE-TIER

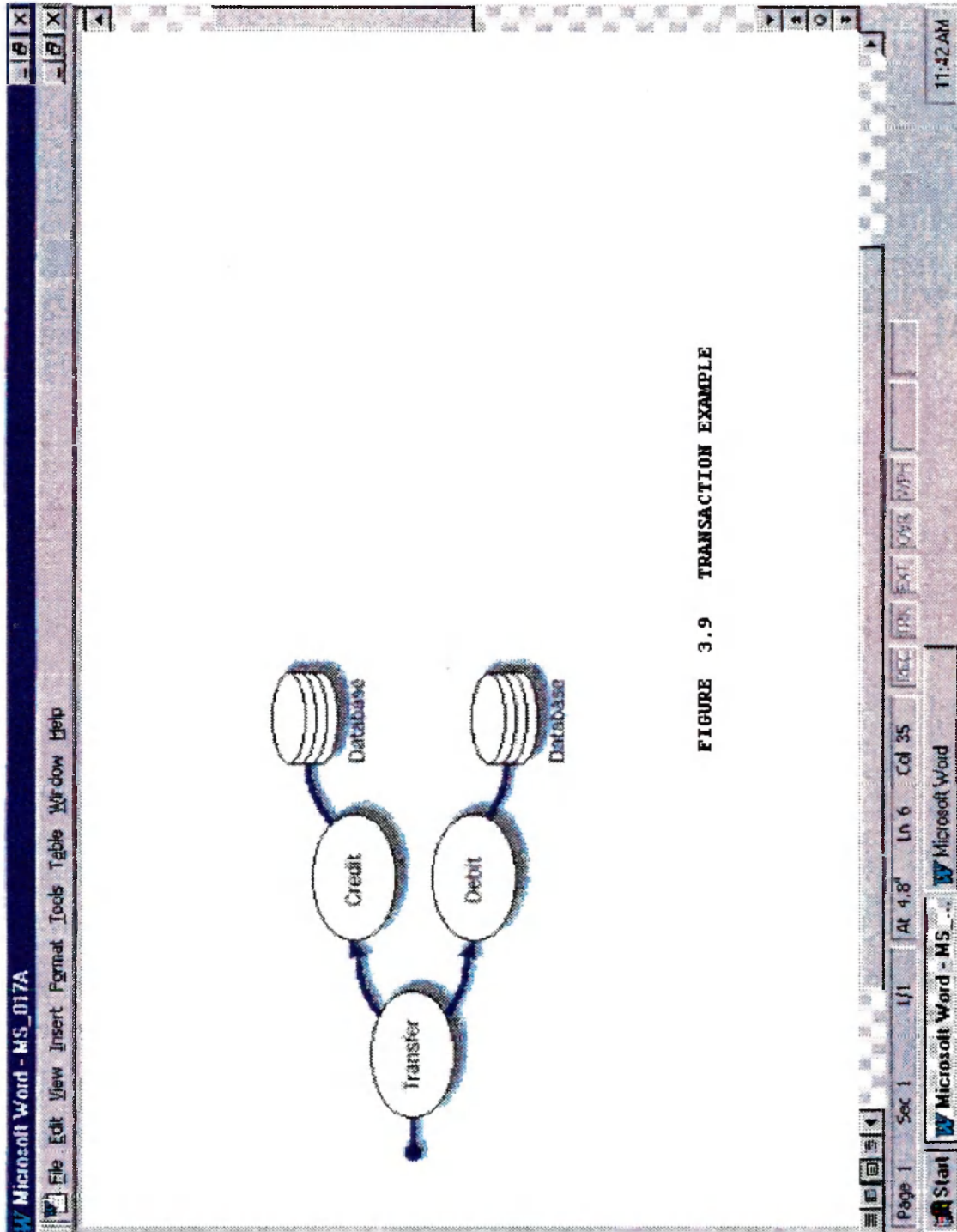


FIGURE 3.9 TRANSACTION EXAMPLE

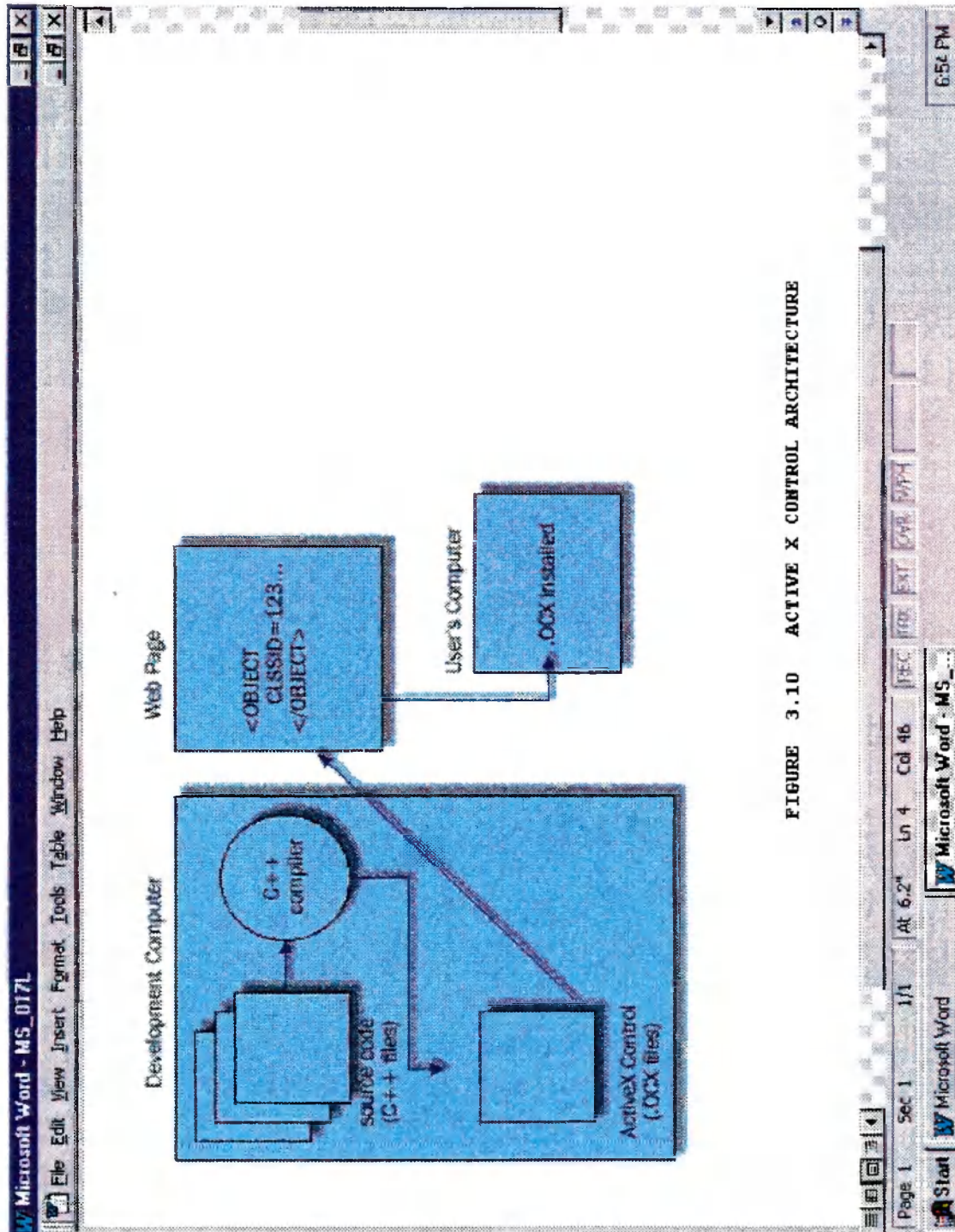


FIGURE 3.10 ACTIVE X CONTROL ARCHITECTURE

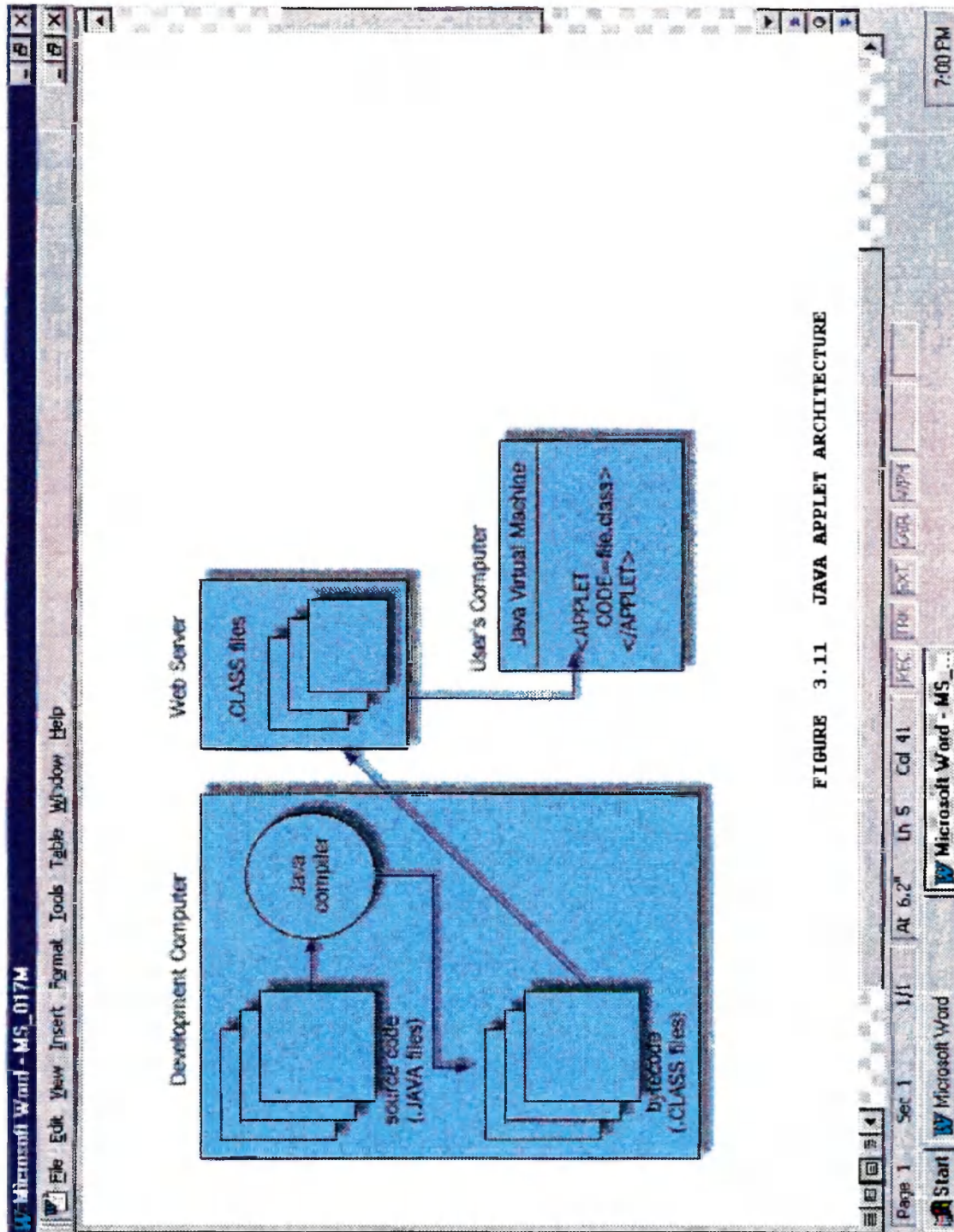


FIGURE 3.11 JAVA APPLET ARCHITECTURE

## Chapter 4

### THE NEW SYSTEM

<i>Chapter Outline</i>	<i>Page</i>
4.1.- Introduction .....	74
4.1.1 The Software Challenge .....	74
4.1.2 Outline of the rest of the chapter .....	74
4.2.- Development of the Prototype .....	75
4.2.1 The Home Page (Figure 4.1) .....	75
4.2.1.1 User Perspective .....	75
4.2.1.2 Technical Overview .....	76
4.2.2 Option A. International Invoice Number (Figure 4.2).....	78
4.2.2.1 User Perspective .....	78
4.2.2.2 Technical Overview .....	78
4.2.3 Option 1. Order Selection Screen (Figure 4.3) .....	80
4.2.3.1 User Perspective .....	80
4.2.3.2 Technical Overview .....	80
4.2.3.3 Screen Selecting Lines (Figure 4.4) .....	81
4.2.3.3.1 User Perspective .....	81
4.2.3.3.2 Technical Overview .....	81
4.2.4 Option 2. Invoice Header Information (Figure 4.5 - 4.9).....	84
4.2.4.1 User Perspective .....	84
4.2.4.2 Technical Overview .....	84
4.2.5 Option 3. Order Removal Screen (Figure 4.10).....	87
4.2.5.1 User Perspective .....	87
4.2.5.2 Technical Overview .....	87
4.2.6 Option 4. Line Selection Screen (Figure 4.12) .....	90
4.2.6.1 User Perspective .....	90
4.2.6.2 Technical Overview .....	90
4.2.6.3 Button Changes (Figure 4.13) .....	91
4.2.6.3.1 User Perspective .....	91
4.2.6.3.2 Technical Overview .....	92
4.2.6.4 Button Add Detail Line (Figure 4.14).....	92
4.2.6.4.1 User Perspective .....	92
4.2.6.4.2 Technical Overview .....	93
4.2.6.5 Button Add Text Line Before (Figure 4.4).....	93
4.2.6.5.1 User Perspective .....	93
4.2.6.5.2 Technical Overview .....	94

4.2.6.6	Buttons Return and Delete (Extract X14)	94
4.2.6.6.1	User Perspective	94
4.2.6.6.2	Technical Overview	94
4.2.7	Option 5. Invoice Text Lines (Figures 4.16 - 4.17)	96
4.2.7.1	User Perspective	96
4.2.7.2	Technical Overview	96
4.2.7.3	Selection Total Lines (Figure 4.18)	97
4.2.7.3.1	User Perspective	97
4.2.7.3.2	Technical Overview	97
4.2.7.4	Selection to Change (Figure 4.19)	98
4.2.7.4.1	User Perspective	98
4.2.7.4.2	Technical Overview	98
4.2.8	Option 6. Invoice Authorization Maint (Figure 4.20 - 4.21)	99
4.2.8.1	User Perspective	99
4.2.8.2	Technical Overview	99
4.2.9	Option 7. Invoice Preview Screen (Figure 4.22 - 4.24)	100
4.2.9.1	User Perspective	100
4.2.9.2	Technical Overview	100
4.2.10	Option 8. Invoice Print (Figure 4.25 - 4.26)	101
4.2.10.1	User Perspective	101
4.2.10.2	Technical Overview	101
4.3.-	Difficulties And Issues Arising When Creating The Prototype	102
4.4.-	Cautionary Guidance For Future Migrators	105
4.5.-	Summary	107

## 4.1 Introduction

### 4.1.1 The Software Challenge<sup>13</sup>

Einstein once said that there must be fundamentally simplified explanations for everything, since God is not capricious or arbitrary. This faith in the underlying simplicity of the artifact of study is a intellectual cornerstone of the natural sciences. Unfortunately, Einstein's notion of underlying simplicity in the natural sciences seldom seems to apply in the sciences of the artificial (the science of man-made systems). Software is abstract and obeys no physical laws, unlike bridges or buildings that obey the laws of physics. [PRIE93] Furthermore, the instinctive feel that humans have for physical artifacts is typically absent when they relate to software. A large software system is characterized by an enormous state space of possibilities, rendering it difficult to understand and appreciate. Simply put, software is hard.

### 4.1.2 Outline of the rest of the chapter

In this chapter this paper will describe the development of the prototype. Then this document will present the Difficulties and Issues arising when the prototype was being created and finally this writing will present some points which will be helpful to future migrators.

---

<sup>13</sup> Shyam R. Chidamber., Metrics for Object Oriented Software Design, Alfred P. Sloan School of Management Ph. D. Thesis, (1997)

## 4.2 Development of the Prototype

In the development of the prototype it will be used the Microsoft tools since Valmont uses this tools. However we want to point out that there are others tools in the market that may be used to fulfil the same goal. It is also important to mention that the prototype's architecture will be based in the Microsoft intranet Architecture and it will use a Three Tier Approach. As we commented in chapter three this Architecture is general and can be implemented using a COBRA or a DCOM approach. Please refer to chapter three and in particular to section 3.6 for a detailed comment about implementation.

The prototype uses tables located in the database server. The tables were mapped from the relational database located in the mainframe to the database server. Some minor revisions to some of the tables (fields) were executed. The revisions were mainly done to adjust field size. Please refer to Appendix A Tables composition.

The next paragraphs describe the implementation of the International Invoice intranet application from two perspectives: the interface and displays screens that appear within the content of the user's decision-support requirements and a technical overview that summarizes the activity that occurs "under the covers" on the intranet and at the server.

### 4.2.1 The Home Page (Figure 4.1)

#### 4.2.1.1 User Perspective

At our company a user Mary Smith arrives at work and from her desktop after signing in the system her user ID and her PASSWORD she accesses a version of the International Invoice home page using Microsoft's Internet Explorer browser (please see figure 4.1). The home page contains information including the company's logo and the company's corporate mission and strategy.

An important piece of this main page is a navigational list located in the lower left part of the screen. This list is similar to a table of contents and will allow the user to navigate to the majority of the pages in the International Invoice Web Site. (I.I. Web Site)

Mary clicks on the International Invoice Number link and she is presented with the screen Invoice and Dealer (please see figure 4.2)

#### **4.2.1.2 Technical Overview**

Mary can employ any of the usual Web browser methods (for instance typing in URLs or using hotlinks or bookmarks) to navigate through the corporate intranet to retrieve specific information.

A User server-based profile, which is initially created by the system administrator, based on user requirements determines the specific content, bookmarks, and sources available to the user. The user can control standard browser settings and configuration options to customize the browser environment based upon personal preferences.

Essentially, the user interface "adapts" to the needs of the individual user based on specifications provided by the user. Since the profile is server based, Mary has the same desktop and security access rights regardless of what PC she is using.

When Mary types the intranet home page URL into the Microsoft Web browser, the browser downloads the associated Web page, an HTML document, from the web server (please see figure 4.1). Technically, this operates the same as accessing any standard page from a browser. Once the home page is downloaded to Mary's browser, she is disconnected from the web server. Note: this is a standard procedure with web browsers.

While some pages are stored on the server as static (preexisting HTML documents) others are created on the fly based upon the user's request for information. The International Invoice home page (figure 4.1) represents a static Web page, while the Invoice and Dealer page (figure 4.2) is an example of a dynamically created Web page because this page presents a selection box with the available International Dealers and since new dealers are added and some dealers are removed this selection box reflects the existing dealers in our Database.

The home page default.htm was developed using a combination of HTML and VB script. This page contains three frames that reference the following three pages:

- i.- Crest.htm: which is the upper left frame. It simply contains the company's logo
- ii.- Links.htm: which is the lower left frame contains a navigational list. This navigation page contain Hyperlinks to most of the pages in the I.I. Web Site. In reality the Hyperlink line (for instance the Hyperlink line 1. Order Selection Screen) is an object, then an event procedure is created for this object and this event procedure performs some logic when the Hyperlink is clicked; for example it may load another page. (please see EXTRACT X0)
- iii.- Miko\_home.htm: which is the right frame is the main informational page for the International Invoice Web site.

#### 4.2.2 Option A. International Invoice Number (Figure 4.2)

##### 4.2.2.1 User Perspective

The user is presented with the screen Invoice and Dealer please see figure 4.2 in this screen Mary is presented with a selection box and five input fields. The selection box presents all the dealers available. The user selects a dealer by clicking its number and then, the five input fields: customer number, sold to, Currency, Rate and Invoice Number are populated. Note that the currency value defaults to United States Dollars, the transfer rate defaults to 1 and the invoice number is populated with the next invoice sequential number. In this screen she has the ability to override the default values for currency and rate of exchange. The user has also the ability to retrieve an Invoice which is already in the system by entering the desired invoice number. After the user has made her selection she clicks the button submit and she is redirected to our home page and the values selected will be captured to be used when she makes use of the navigational list.

Note: In the case that a user may try to access a different page before accessing the screen Invoice and Dealer she will be redirected to this screen to enter the particular data for this run and after she clicks submit she will be transported to her initial desired page.

##### 4.2.2.2 Technical Overview

The file encharged of the functionality of the screen Invoice and Dealer is the Active Serve Page Profile\_asp. (please see EXTRACT X1) This file uses client side script, and server side script as well; the details follows:

- i.- The profile\_asp is a reentrant page, it means that the response is the same page. This page contains a self-targeting form and server-side code that allows it to both collect information and display summary information.

ii.- The profile\_asp page collects information from the HTML form and to make the form data available to other pages in the I.I. Web Site it stores the data in Session Variables.

iii.- To achieve the redirection of users to the Invoice and Dealer page a script was added on the Session\_onStart of the file Global\_asa (please see EXTRACT X2) that redirects the users to the file profile\_asp if they attempt to start the Session with a different page.

iv.- To populate the dealers selection box, in our profile\_asp page, we used server side script code and created an instance of a component by using the ASP Server Object's CreateObject method and called the VCUSTall component. VCUSTall (please see EXTRACT X3) is a method which creates a connection to the target ODBC database using a connection string and retrieves the current dealers related information. Then we used a variant array to store the information passed from the server to the client. In the same fashion, by using the component MIKO\_IAA\_PULL (please see EXTRACT X3) we retrieve the next sequential invoice number.

v.- The dealers selection box uses an event handler (clicked event) and used properties and methods for script objects in response to the user's selection to populate the input data controls. (please see EXTRACT X1)

### 4.2.3 Option 1. Order Selection Screen (Figure 4.3)

#### 4.2.3.1 User Perspective

When Mary clicks Selection 1 she will be guided to the screen Order Selection Screen. (please see figure 4.3) This screen is used to select orders for a given dealer. This screen shows, in ascending order, the orders that have at least one line with an available quantity and the number of lines that the order owns. The user is presented with the first 6 lines; if she desires to see the next six orders she will click the next button.

The user can select the desired order by clicking the order number (Order number click to select.)

#### 4.2.3.2 Technical Overview

The Order Selection link that Mary selected is actually, a reference to the ASP page Transcript\_asp. This page manages the functionality of this selection link by working in collaboration with the component R1OR\_SEL\_VB the Bid\_asp and the Next\_asp active server pages. The details follows:

- i.- When the page transcript\_asp is instantiated, it retrieves the Session variables dealer number and invoice number.
- ii.- Transcript\_asp uses server side script to create an instance of a component and call the R1OR\_SEL\_VB component passing it a dealer number.
- iii.- R1OR\_SEL\_VB is a method which creates a connection to the target ODBC database. This method is encharged of the implementation of business rules for our domain and is entrusted of the generation of a group of records that are transferred to the requester (client.)

iv.- The data passed from the server is stored in a variant array, then we create an HTML table that has two columns and by using a combination of HTML and Client Server side script the data is presented to the user in such a way that the Order Number is a Hyperlink.

v.- Please see figure 4.3 for the visual presentation of the Order Selection Screen and please see EXTRACT X4 for a shot of the code that creates that view.

#### **4.2.3.3 Screen Selecting Lines (Figure 4.4)**

##### **4.2.3.3.1 User Perspective**

When Mary clicks the desired order, She is presented with the screen Selecting Lines. (please see figure 4.4) This screen will show all the lines (in ascending order) that belong to that particular order. In this screen Mary can make three selections a.- She can select just one line by clicking the desired line number. b.- She may select a range of lines by filling the input boxes Range Start and Range End and by clicking the button Range. c.- She can select all the lines of the order by clicking the button All. After Mary makes her request a screen telling her that her request was successfully completed is provided.

##### **4.2.3.3.2 Technical Overview**

The order number hyperlink is really a citation to the asp page Bid\_asp. This page is entrusted of generating the screen selecting lines. Please see figure 4.4. To accomplish its task Bid\_asp works in collaboration with the component R1OR\_SEL2A\_VB and with the active server pages AddVB\_asp, AddVBRANGE\_asp, and AddVBALL\_asp. The details follow:

i.- When the page Bid\_asp is instantiated it requests the value of the order clicked and it also retrieves from a session variable the value of the dealer number.

ii.- Bid\_asp uses server side script to create an instance of a component and call the R1OR\_SEL2A\_VB component passing it a dealer number and the order clicked. (please see EXTRACT X4)

iii.- R1OR\_SEL2A\_VB is a method which creates a connection to the target ODBC database. This method is entrusted of the implementation of business rules for our domain and is encharged of the generation of a group of records that are transferred to the requester (client.)

iv.- The data passed from the server is stored in a variant array, then by using a combination of HTML and server side script the passed data is presented in a four frame two columns table ordered in ascending fashion by line number and where the line number is a hyperlink to the AddVB\_asp file. (please see EXTRACT X5)

v.- Bid\_asp uses client-side script so that when the user click the buttons Range or the button All; control is transferred to the AddVBRANGE\_asp, or AddVBALL\_asp.

vi.- When the user Selects one line, by clicking the hyperlink line number, control is passed to the active server page AddVB\_asp. This page uses server-side script to create an instance of a component and to call the Vbline\_1 component passing it the line number, the customer number, the order number and the invoice number. The passed parameters are obtained either by requesting them or by retrieving them from session variables.

vii.- Since the selection of a line is a transaction that changes the state of the database. It must conform to the ACID properties. The transaction can be achieved by using components or by using a stored procedure. In this case the transaction will be achieved by using a component; later it will be presented a similar case but the transaction will be achieved by using a stored procedure.

More about this situation when this paper itemizes the difficulties and issues arising in this project.

viii.- Because the selection of a line is achieved by updating two tables this action must be accomplished or the whole transaction must be rolled back. Vbline\_1 (a component) carries this challenge by using the properties BeginTrans, Execute, CommitTrans, and RollbackTrans of the conn object. (please see EXTRACT X6)

ix.- A similar process is executed when the user chooses to select a range of lines or to select all lines.

x.- Please see figure 4.4 for the visual presentation of the Selecting Lines Screen and please see EXTRACT X5 for a shot of the code that creates that view.

#### 4.2.4 Option 2. Invoice Header Information (Figure 4.5 - 4.9)

##### 4.2.4.1 User Perspective

When Mary clicks selection 2 Invoice Header Information the system can display two different screens depending on the information that she provided in the initial screen. a.- Case one, when Mary is retrieving a Invoice which is already in the system she will be presented with a screen for update Please see figure 4.5. in this screen she will be able to post her desired changes and then she will click the button Update to transfer her changes to the Database. b.- Case two, when Mary is generating a new Invoice she will be presented with a screen which let her know that the Invoice number is not in the system. Please see figure 4.6, in this screen when she clicks the button New to confirm that she is creating a new Invoice; she will be transported to the screen for new Invoice Please see figure 4.7. In this screen the system is providing the Invoice Number, the dealer number and many other values that the system has already captured (These fields are shown with green background.) then Mary clicks the button Insert and the new record will be added to the Database.

Note: The fields Code terms, and Employee Authorization are drop down lists where the user can make her selections. Please see figures 4.8 and 4.9

##### 4.2.4.2 Technical Overview

The Invoice Header Information hyperlink is really an allusion to the active server page Class77Action\_asp. The main task of this page is to execute the commands that the user requested and to coordinate with the active server page Class77Form\_asp the presentation of the different screens for the user. To accomplish their task these files work in collaboration with the active server pages Rau\_Head\_Action\_asp, and Raul\_Head\_Form\_asp. The details follow:

- i.- The presentation of the screens Update record and the New record is handled by the file Class77Form\_asp. In the case that the record is already in the system a Recordset object (rsRau9555A1SQLQuery) for that particular invoice number is created. (please see EXTRACT X7) This Recordset object holds the information belonging to that particular invoice number.
- ii.- In the case that the record is not in the system a new Recordset object with the new invoice number is generated.
- iii.- Class77Action\_asp uses the methods of the Recordset object to handle the modifications to the different fields and the insertions or the updates to the target database. Some of the methods that this file uses are AddNew (to add a new record) , Update (to update an existing record) , Requery (to update the data in the recordset by re-executing the query on which the object is based). When the prototype was being created an unusual situation surfaced the methods above mentioned worked properly when the target database was SQL server. However when we used as our target database Oracle server none of them behaved properly. I was forced to write this methods as a components in such a way that the modifications in the database were executed by the components. More about this situation when this paper itemizes the difficulties and issues arising in this project.
- iv.- Class77Action\_asp uses server side script to create an instance of a component and call the Vbline\_SOS\_B component passing it a sql string. (please see EXTRACT X8) The active server page is entrusted of generating the sql string that can be an update a record string or an add a new record string. The only job of the component is to execute the sql string against the target database. In the case of adding a new record the Invoice Number storage is incremented by one in order to be ready to show (in the Invoice and Dealer screen) the next sequential Invoice Number. The component encharged of the increment is MIKO\_IJA\_UPD. This Invoice Number storage is retrieved when the

Profile\_asp page is implemented. The component entrusted of the retrieve is MIKO\_IIA\_PULL (please see EXTRACT X1).

v.- The implementation of the two drop down lists for the fields Code terms, and Employee Authorization are very similar. Both use a component to retrieve from the Target server database the data. Then the data is passed from the server to the client. Then the client stores the data in a variant array and by using HTML code and a SELECT container the data is displayed as a selection list.

### 4.2.5 Option 3. Order Removal Screen (Figure 4.10)

#### 4.2.5.1 User Perspective

When Mary clicks selection 3 Order Removal Screen the system presents her with the screen Order Removal Screen. Please see figure 4.10, In this screen the orders assigned to an International Invoice Number will be displayed. This screen allows Mary to remove any order assigned to a particular International Invoice Number. To carry on the removal she needs to click the Order number to be removed. (Order number click to Remove)

#### 4.2.5.2 Technical Overview

The Order Removal Screen hyperlink is actually a reference to the active server page Maria\_3\_asp. This page with the help of the component vCallStoredProc and the active server page Bid3\_new\_asp handles the removal of an order from an Invoice. The removal of an order is a transaction that changes the state of the database and since that transaction must conform to the ACID properties the Order removal screen option will allow this paper to present the achievement of those properties by using a stored procedure, the details follows:

- i.- Maria\_3\_asp handles the presentation of the Order Removal Screen. To retrieve the orders belonging to a particular invoice it calls the component vCallStoredProc passing it the invoice number. Then when the requested information is returned it is presented to the client by using a combination of a server-side script and HTML in such a way that the Order number is a hyperlink to the active server page Bid3\_new\_asp. (please see EXTRACT X9)
- ii.- vCallStoredProc is a method which creates a connection to the target ODBC database. This method is assigned of the implementation of business rules for our domain. Its main task is to retrieve and to transfer the selected orders to the requester (client).

iii.- When the user clicks an order hyperlink to remove the order number, control is passed to the active server page Bid3\_new\_asp. This page uses server-side script to create an instance of a component and to call the IID\_IIR3 component passing it the invoice number, the customer number, and the order number. The passed parameters are obtained either by requesting them or by retrieving them from session variables.

iv.- The removal of an order is a transaction that changes the state of our database therefore it must conform to the ACID properties. The transaction can be achieved by using components or by using stored procedures. In this case the transaction will be achieved by using a stored procedure; earlier it was presented a similar case but the transaction was achieved by using a component (please see EXTRACT X6). More about this situation when this paper itemizes the difficulties and issues arising in this project.

v.- Because the removal of an order is attained by updating two tables this action must be accomplished or the whole transaction must be rolled back. IIR3\_1A (a stored procedure) carries this challenge. This procedure accepts five parameters and then by using BEGIN TRAN it executes the changes in the two tables, then if no errors are detected the whole transaction is committed (COMMIT TRAN.) In the case that errors are detected the transaction is rolled back (ROLLBACK TRAN.) (please see EXTRACT X10.)

vi.- The EXTRACT X10 shows the three tier architecture being used. Here at one end the active server page (Bid3\_new\_asp) deals with the presentation tier or user interface. When the presentation needs to carry out an action in the server it requests to the business tier that the action be executed (by calling the IID\_IIR3 component). Then in the middle the component IID\_IIR3 deals with the processing of the business rules (logic tier.) Then when the business layer needs to carry out an action against the database it requests to the data tier that the action be executed (by calling the IIR3\_1A stored procedure.) Finally at the

other end in the data server where the stored procedure resides the actions against the database are implemented.

#### 4.2.6 Option 4. Line Selection Screen (Figure 4.12)

##### 4.2.6.1 User Perspective

When Mary clicks selection 4, Line Selection Screen the system presents her with all the line orders belonging to a particular international invoice. Please see figure 4.11. In this screen the user can manipulate each one of the detail lines; for instance the quantity shipped can be changed and the column total amount for that particular line will be recalculated. To introduce the alterations to a particular line Mary must select the line by clicking in the line number (Line Choice) after she selects the line, she will be presented with a screen for that particular choice. Please see figure 4.12. In this screen, Mary is presented with five selection buttons. The button "Changes" will allow her to make changes to the price, to the part number and discount of the selected line. The button "Add detail line" will allow her to input a new line. The button "Add text line before" will allow her to insert a text line before the detail line. The button "Return for future selection" will allow her to remove a detail line from the international invoice number therefore making the line values available to be selected by a different international invoice. The button "Delete (will never be invoiced)" will delete the line selected. To carry any of the five options above mentioned Mary needs to click the appropriate button.

##### 4.2.6.2 Technical Overview

The Line Selection Screen hyperlink is really an allusion to the active server page Raul78\_4\_asp. This page with the help of the component Raul78\_4StoProc handles the retrieve and the presentation of the detail lines belonging to an international invoice. Then this page with the aid of the file Bid4\_asp presents the line selected for further processing. The details follow:

- i.- Raul78\_4\_asp handles the presentation of the Line Selection Screen. (figure 4.11) the retrieve of lines is done by calling the component Raul78\_4StoProc passing it the invoice number. The lines are presented in such a way that the line number (Line Choice) is a hyperlink.
- ii.- Raul78\_4StoProc is a method which creates a connection to the target ODBC database. This method is entrusted of the implementation of business rules for our domain. Its main task is to retrieve and to transfer the selected lines to the requester (client).
- iii.- When Mary clicks the line number hyperlink (line choice) to select the line, control is passed to the file Bid4\_asp. (please see EXTRACT X11) This page uses server-side script and HTML to present the screen line selected. (figure 4.12) This file also uses client-side script to capture the on click events of the respective buttons.

#### **4.2.6.3 Button Changes (Figure 4.13)**

##### **4.2.6.3.1 User Perspective**

When Mary wants to make a change to the selected line she clicks the button Changes then she is transported to the screen Option 4\_change. (Please see figure 4.13) in this screen Mary is presented with the Part Number, Price, Discount, Quantity shipped, the description and the flag for price to be charged or to be included. In this screen she is able to update any of those fields. After she made their changes she clicks submit to store the changes in the Database. And then Mary is transported to the line selection screen (figure 4.11) where she can observe that the changes she just made are presented (took effect.) Note that all the updateable fields have a green background, this is to make an analogy with the CICS screens in the mainframe; the CICS screens used a green color for the

updateable fields. More about this analogy when this paper itemizes the difficulties and issues arising in this project.

#### **4.2.6.3.2 Technical Overview**

The button Changes is a reference to the active server page RAU4\_CHANGES\_ASP this page handles the presentation of the screen Option 4\_change. (Please see figure 4.13) This asp file with the help of the component VB\_4\_31 is encharged of executing the implementation of the changes requested by the user. The details follow:

- i.- RAU4\_CHANGES\_ASP is a reentrant page, it means that the response is the same page (please see EXTRACT X12). This page contains a self-targeting form and server-side code that allows both to display and to collect information.
- ii.- The form using a combination of server side script and HTML presents the data. (Please see figure 4.13) The user makes her changes and clicks the button submit. The submit button transmits the data to the reentrant page.
- iii.- The reentrant page, validates the data and calls the component VB\_4\_31 passing the required information for the changes to be implemented.

#### **4.2.6.4 Button Add Detail Line (Figure 4.14)**

##### **4.2.6.4.1 User Perspective**

When Mary wants to add a detail line she clicks the button Add detail line then she is transported to the screen Add a Detail Line. (Please see figure 4.14) In this screen Mary is presented with nine input boxes. After she fills the input boxes Mary clicks submit to store the detail line in the Database, then Mary is transported to the line selection screen (figure 4.11) where she can observe that the line just inserted is presented.

#### 4.2.6.4.2 Technical Overview

The button Add detail line is a reference to the active server page Rau4\_AddDline\_asp this page handles the presentation of the screen Option Add a Detail Line. (Please see figure 4.14) Rau4\_AddDline\_asp with the help of the components RAU4\_CHK\_D\_R and VB\_4\_32 are entrusted of implementing the insertion of the detail line. The details follow:

- i.- Rau4\_AddDline\_asp is a reentrant page. It validates the data and call the component RAU4\_CHK\_D\_R.
- ii.- The component RAU4\_CHK\_D\_R is encharged of checking if the line being added already exist. In the case that the line already exist a message is presented to the user saying that "We are adding lines not Editing".
- iii.- The component VB\_4\_32 is entrusted of adding the line to the Database

#### 4.2.6.5 Button Add Text Line Before (Figure 4.4)

##### 4.2.6.5.1 User Perspective

When Mary wants to Add a text line before the line selected she clicks the button "Add text line before" then Mary is transported to the screen Text Line. (Please see figure 4.15) In this screen the user is presented with a text area container. After she fills the container Mary can click either the button "Save" to store the text line in the Database or the button "Reset" to return the textarea to its original value. Next Mary is transported to the line selection screen. (figure 4.11) The user can later edit or delete the text line by using a similar procedure.

#### 4.2.6.5.1 Technical Overview

The button "Add text line before" is a reference to the active server page New\_RAU44A1\_ASP this page handles the presentation of the screen Option Text Line. (Please see figure 4.15) New\_RAU44A1\_ASP with the help of the components VB\_41\_4txtUP and VB\_41\_4txt are entrusted of implementing the insertion of the text line. The details follow:

- i.- New\_RAU44A1\_ASP is a reentrant page. The asp file calls the component VB\_41\_4txt to retrieve the current value of the Text Line.
- ii.- New\_RAU44A1\_ASP using a mix of HTML and vb script is assigned of presenting the screen Text Line. (figure 4.15) (please see EXTRACT X13)
- iii.- The component VB\_41\_4txtUP is encharged of adding the line to the Database.

#### 4.2.6.6 Buttons Return and Delete (Extract X14)

##### 4.2.6.6.1 User Perspective

When Mary wants that the line selected be Returned for future selection she clicks the button "Return for future selection" then a screen confirmation appears saying that her request was accomplished. By the same token when Mary wants to delete the line selected she clicks the button "Delete (will never be invoiced)" then a screen confirmation appears saying that her request was accomplished.

##### 4.2.6.6.2 Technical Overview

The button "Return for future selection" is a reference to the active server page ADDFOUR\_ASP this page by using a select case calls the required component

either DR4DELB or DR4RETVB to accomplish the user requested action.  
(please see EXTRACT X14)

#### 4.2.7 Option 5. Invoice Text Lines (Figures 4.16 - 4.17)

##### 4.2.7.1 User Perspective

When Mary clicks selection 5, Invoice Text Lines the system presents her with the screen Invoice Text Lines. This screen is a menu which allows the user to select prefix, post, change, or total text lines for international invoicing. Please see figure 4.16 In this menu Mary can click any of the options for instance when she clicks the option "Prefix Lines (Lines Before Detail Lines)" she is transported to the screen Prefix Lines. Please see figure 4.17 In this screen the user is presented with a text area container. After she fills the container, Mary can click either the button "Save" to store the text line in the Database or the button "Reset" to return the textarea to its original value. Next Mary is transported to the Invoice Text Lines menu. (figure 4.16) The user can later edit or delete the text line by using a similar procedure. The options "Post Lines (Lines After Detail Before Totals)" and "Post Lines (Lines After Totals Before Auth)" are similar to the option "Prefix Lines" and their functionality is alike.

##### 4.2.7.2 Technical Overview

The selection 5, Invoice Text Lines hyperlink is really a citation to the active server page RAU\_55HTM\_ASP. This asp file is a menu and is entrusted of presenting the six options which are hyperlinks to a corresponding asp files (please see EXTRACT X15.) The three options "Prefix Lines (Lines Before Detail Lines)", "Post Lines (Lines After Detail Before Totals)", and "Post Lines (Lines After Totals Before Auth)" are implemented in the same fashion as the option "Add text line before". Its technical description was given in Technical Overview 3.

### 4.2.7.3 Selection Total Lines (Figure 4.18)

#### 4.2.7.3.1 User Perspective

When Mary clicks the option Total Lines the system presents her with the screen Text Total Lines. Please see figure 4.18 This screen is used to enter total lines for international invoicing. The first line is reserved for the Sub-Total. (The total of the detail lines) The last line is saved for the Grand Total, or the Total of all the detail lines and the extra charges (freight, documents, etc.) The middle lines are reserved for miscellaneous charges like freight, insurance, documentation etc. The system will figure the Sub Total and the Grand Total. The buttons "Save" and "Reset" are standard options and they either store the screen information in the Database or return the screen to its original value. When Mary finish her work in screen Text Total Lines she is transported to the Invoice Text Lines menu. (figure 4.16)

#### 4.2.7.3.2 Technical Overview

The hyperlink "Total Lines" is a reference to the active server page NEW\_RAU55B1\_ASP this page handles the presentation of the screen Option Text Total Lines. (Please see figure 4.18) This asp file with the help of the components VB\_41\_52, VB\_41\_52UPEQ, and MAKE\_NEW\_555 are assigned of implementing the functionality of the figure 4.18. The details follow:

- i.- NEW\_RAU55B1\_ASP is a reentrant page. This asp file calls the component VB\_41\_52 passing it the invoice number to retrieve the current values of the Screen Text Total Lines. Next the asp file using a mix of HTML and vb script is encharged of presenting the screen Text Total Lines (figure 4.18) (please see EXTRACT X16)
- ii.- The component VB\_41\_52 is a method which creates a connection to the target ODBC database. Its main task is to retrieve and to calculate the Sub Total

value of the detail lines. One of its functions is also to transfer the selected information to the requester (client). To achieve its objective VB\_41\_52 uses the help of the function UPD\_TOTAL\_9999. This function is entrusted of creating the necessary lines in the respective tables when the creation of lines is necessary.

iii.- The component VB\_41\_52UPEQ is assigned of updating the information in the Database. To carry on its task it uses the help of the function IsEqual, RestoreNull, and ConvertToString. These functions as their name suggests deal with the handling of the passed parameters. (please see EXTRACT X17)

iv.- The component MAKE\_NEW\_555 is encharged of inserting the middle lines(miscellaneous charges) in the Database. (please see EXTRACT X17)

#### **4.2.7.4 Selection to Change (Figure 4.19)**

##### **4.2.7.4.1 User Perspective**

When Mary clicks the option To Change (For Valmont Industries, Inc.) the system presents her with the screen Auth Text Line. Please see figure 4.19 This screen is used to update the text line for International Invoicing Authorization Line. When Mary finish her update she clicks submit and then she is transported to the Invoice Text Lines Menu. (figure 4.16) The option to Change Subtotal Verbage behaves in similar fashion as the option To Change (For Valmont Industries, Inc.)

##### **4.2.7.4.2 Technical Overview**

The hyperlink "To Change (For Valmont Industries, Inc.)" is a reference to the active server page RAU55E\_ASP this page handles the presentation of the screen To Change (For Valmont Industries, Inc.), figure 4.19, This asp file with the help of components handles the retrieve and the update of the record.

#### 4.2.8 Option 6. Invoice Authorization Maint (Figure 4.20 - 4.21)

##### 4.2.8.1 User Perspective

When Mary clicks selection 6, Invoice Authorization Maint, the system presents her with the Maintenance Screen. This view presents all the authorized employees. Please see figure 4.20. In this screen the user is allowed to make changes to any of the authorized employees. To execute her changes Mary will click the desired employee ID then she is presented with the screen Maintenance Screen (2). Please see figure 4.21 This screen presents her with the detail information of the employee selected. In this panel she has the option to update this particular record by making her modifications and pressing the button Changes or she may delete the record by pressing the button Delete (Employee) In the case that she wants to add a new employee she will click the button Add Employee and a screen for entering the new data will be presented. After she finish her maintenance a screen telling her that her request was successfully completed is provided.

##### 4.2.8.2 Technical Overview

The selection 6, Invoice Authorization Maint hyperlink is actually a reference to the active server page MENU\_6\_ASP. This asp file with the help of the component VEMPL retrieves the employees records and presents them to the user (figure 4.20) in such a way that the employee number is a hyperlink to the file BID6A\_ASP. This asp file presents the record selected and three button options for Changes, Delete and Add. The details of the implementation of button's actions is similar to the technical description given in the Technical Overview 4.2.6.3.

#### **4.2.9 Option 7. Invoice Preview Screen (Figure 4.22 - 4.24)**

##### **4.2.9.1 User Perspective**

When Mary clicks selection 7, Invoice Preview Screen, the system presents her with a view only screen that shows her the International Invoice that she just created. This Preview Screen is specially formatted for the easy identification of the different parts that compose an International Invoice. Please see figures 4.22, 4.23 and 4.24.

##### **4.2.9.2 Technical Overview**

The selection 7, Invoice Preview Screen hyperlink is really an allusion to the active server page RAUL77\_ASP. This asp file makes use of the different components presented in the previous technical overviews and with a mix of HTML and vb script formats the screen for the user.

#### **4.2.10 Option 8. Invoice Print (Figure 4.25 - 4.26)**

##### **4.2.10.1 User Perspective**

When Mary clicks selection 8, Invoice Print, the system presents her with a formatted view ready to be printed. In this screen she will be able to generate a hard copy of the International Invoice. Please see figures 4.25, 4.26, for an example of a International Invoice hard copy.

##### **4.2.10.2 Technical Overview**

The selection 8, Invoice Print is actually an allusion to the active server page RH3\_ASP. This asp file makes use of the different components presented in the previous technical overviews and with a mix of HTML and vb script formats the screen and makes it ready for the printer. Please see figures 4.25, 4.26, for an example of an International Invoice hard copy.

### 4.3.- Difficulties And Issues Arising When Creating The Prototype

There were many issues that surfaced when the prototype was being created. The most important are:

i.- The software world is a rapidly maturing domain. The development of the prototype was initiated using Microsoft (Visual Interdev 1.0). Then we upgraded to the new release (Visual Interdev 6.0). Some modifications to the source code were necessary to transition from the old release to the new one.

ii.- Analogy with the CICS screens in the mainframe. In the CICS screens the fields being pulled from the database (Option 2 Invoice Header Information) have a green background. The prototype presents the same fields with a similar background this is to make an analogy with the CICS screens in the mainframe. The replication of the mainframe's presentation screens will allow the end user to work in a familiar environment and therefore, no important user retraining will be necessary.

\* iii.- Transactions in our Database. The prototype has the possibility to accomplish a transaction either by using a stored procedure or by using a component. When we say using a stored procedure it means that the stored procedure is encharged of implementing the transaction. By the same token when we say using a component it means that the component is encharged of implementing the transaction. In the development of the prototype we achieved the transactions by using in one case a stored procedure and in the other case a component. But when we needed to move the code from the test machine to the production machine the move was less complicated when we used a component since the only change that we needed to make was the change of the line that refers to the target database. However, when we used a stored procedure we needed to create the stored procedure in the target database. This situation was very important since we were using a SQL server 6.0 database in

the test machine; and the database being used in the production machine was ORACLE server 7.4

iv.- Methods work with SQL not with ORACLE. Some methods worked properly when the target database was SQL server. However when the target database was Oracle server they generate errors. This case surfaced when the active server page Class77Action.asp used the methods of the Recordset object AddNew, Update, Requery. The methods above mentioned worked properly when the target database was SQL server. However when we used as our target database Oracle server none of them behaved properly. Microsoft technical support was communicated regarding this issue; they mentioned that a similar case with other user but with the method Delete was being examined. In order to overcome this challenge the prototype was forced to use these methods as a components in such a way that the modifications in the database were executed by the components.

v.- Clients using Internet Explorer 4.0 and Internet Explorer 5.0. Although Internet Explorer is supposed to be upward compatible. Clients using the new version of Internet Explorer (IE 5.0) were having some errors. Those errors did not surface when the client accessed the same screens using (IE 4.0). Microsoft technical support was communicated regarding this issue they mentioned that a IE 5.0 and IE 4.0 use different engines and that some incompatibilities were being reported. The solution was to analyze the piece of code generating the errors make the necessary corrections in such a way that it will work properly with IE 5.0 and at the same time verify that the modified code still work properly with IE 4.0.

vi.- Page Break Styles supported only with block elements. When the prototype was printing the Invoices the page breaks were not having the correct behavior. Microsoft technical support was communicated regarding this issue; they mentioned that the example that Internet client SDK contains (example of how

to use the page-break after to force Internet Explorer to insert a page break when printing an HTML document) is incorrect because it uses a BR tag, which is not a block element. (Solution: Replace the <BR> tag and BR.page style definition with a <P> tag and P.page style to achieve the correct behavior).

vii.- The users requested additional functionality in the new system. That additional functionality was not being provided by the legacy mainframe application.

#### 4.4.- Cautionary Guidance For Future Migrators

There were many issues that surfaced when the prototype was being created. These issues and their solutions allow this paper to provide some guidelines that future migrators may use to avoid some of the challenges that the development of the prototype faced. It is also important to mention that some of the specifications that this document is presenting were found in the literature reviewed. Then later those specifications were proven right when the prototype was being created. The following is a list that a future migrator may want to take in consideration:

i.- Understand the legacy system. [HORO98] Understand what the legacy system does. Understand how the legacy system accomplish its job. This task may not be easy, since usually legacy systems lack a good documentation. However the effort spend<sup>ed</sup> understanding the old system will be necessary to the succesful completion of the migration.

ii.- Deciding what will be executed on the client side. Deciding what will be executed on the server side. Deciding what communication between client and server will occur. Deciding what browsers the clients will use as a user interface. In the case of the prototype's development since we were using the Microsoft tools the communication between client and server was achieved using COM (DCOM). The user interface for the clients was provided by Internet Explorer 4.0. Also we used a Business Services Model (three tier model). However other alternatives are available. Please, refer to section 3.4 and 3.5 for a discussion about alternatives.

iii.- The old User Interface is going to be replicated or a brand new user iteface is going to be created. Since in the case of the prototype's development one of the goals was to make a change as transparent as possible so no important user

retraining will be necessary. The prototype replicated the legacy user interface to give the users a familiar interface.

iv.- Be ready to call the provider of the software that the new system is using since many of the features will not work as they are advertised or the provider documentation may be in error. In any case be ready to create an alternative solution to continue the development.

v.- The software world is a rapidly maturing domain. There is a chance that the software being used to create the new system will be upgraded. Make sure to allocate a reasonable amount of time for the transition. Since, although the software's producers will assure that their new release is upward compatible chances are that many pieces of code may need to be modified to work with the upgraded software.

vi.- You must make no functional adaptations or enhancements. If adaptations and enhancements are required they should be done in a follow up project. [SNEE95] The users may want to use this opportunity (a new system development) to add new features to the system however the adding of new features will impact negatively in the creation of the new system since it will make more difficult to prove the correctness of the transformation.

vii.- Do development and unit testing in a desktop workstation. Perform as much of the development work as possible independent of the target machine (avoid bottlenecks). [SNEE95] This advice taken from the literature reviewed was one of the most important that the development of the prototype put to practice. The development of the prototype was executed in a stand alone machine and the work was not interrupted when the intranet network was not operative.

viii.- Once the new system is created, traditional issues about its architecture will arise: performance, reliability, and security. [HORO98] In the case of the

prototype for user Security it uses a User server-based profile for details please see section 4.2.1.2. For the part of Reliability the outcome of the test results of the prototype corresponded 100% with the test results of the original legacy system in the mainframe. for the part of performance the new system accomplishment matches the original legacy system and it is expected that with the ongoing efforts of Tunning the Servers the new system will improve its performance.

**Summary:**

In this chapter this paper described the development of the prototype. The new system was created using a Three Tier Approach and the Microsoft tools. The description of the new system was done from two perspectives a user perspective and a technical perspective. Next, this paper presented some issues that arised while the new system was being created. Finally this document presented some helpful hints for future migrators.




FIGURE 4.1 IIMM'S HOME PAGE

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address [http://valmet/IAU\\_2A/default.htm](http://valmet/IAU_2A/default.htm) Links



# Invoice and Dealer

Enter Dealer, Currency, Exchange Rate and Invoice

International Invoicing

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

00008086	Cust Number:	00008100
00008087	Sold To:	VALMONT INT'L CORP.
00008088	Currency:	USD
00008094	Rate:	1
00008100	Invoice Number:	100007
00008102		
00008103		
	Submit	

Start International Invoice Microsoft Word - Document1 Local intranet zone 2:17 PM

FIGURE 4.2 INVOICE AND DEALER

\*NOTE WE CAN MAKE AN ORDER DELETION SCREEN

FOR CANCELLED  
REVISED

International Invoicing

1. [International Invoice Number](#)

2. [Invoice Header Information](#)

3. [Order Removal Screen](#)

4. [Line Selection Screen](#)

5. [Invoice Text Lines](#)

6. [Invoice Authorization Maint](#)

7. [Invoice Preview Screen](#)

8. [Invoice Print](#)

9. [Invoice Air Freight](#)

Order Selection Screen

ID : DEALER : 008058 DEALER NAME : INTL INV : MIK621

Order Number Click to Select	Number of lines <sup>1</sup>
0102094	2
0102498	25
0102794	3
0102798	14
0102896	3
0103294	6
<a href="#">next</a>	

Done

Start International Invoice

5:49 PM

FIGURE 4.3 ORDER SELECTION SCREEN

**Selecting Lines**

ID : DEALER : 008058 DEALER NAME : INTL INV : MIK621

Click Line	Order #	02400	0102498	03000	0102498
00100	0102498	02500	0102498	03100	0102498
00200	0102498	02600	0102498	03200	0102498
00300	0102498	02700	0102498	03300	0102498
02000	0102498	02800	0102498	03400	0102498
02100	0102498	02801	0102498	03500	0102498
02200	0102498	02802	0102498	03600	0102498
02300	0102498	02900	0102498	03700	0102498

Range Start: Range End:

&R Range &A All

Back to Order Selection Screen 1

International Invoicing

- ✓ A. International Invoice Number
- ✓ 1. Order Selection Screen
- ✓ 2. Invoice Header Information
- ✓ 3. Order Removal Screen
- ✓ 4. Line Selection Screen
- ✓ 5. Invoice Text Lines
- ✓ 6. Invoice Authorization Maint.
- ✓ 7. Invoice Preview Screen
- ✓ 8. Invoice Print
- ✓ 9. Invoice Air Freight


#### FIGURE 4.4 SELECTING LINES

International Invoicing - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://nismab2a/StateU/DEFAULT.HTM> Links



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Header Information

INTL INVOICE NUMBER IT WILL WORK : MIK621

Update New

The record

Invoice **mlk621**

Dealer **008562**

Our Ord. No.

Currency Type **USD**

Ship To:

VALLEY-M S.R.L.

STR. A. RUSSO 1, CA 206

AD LINE2

CHISINAU

REP. OF MOSTRATAN

Sold To:

VALLEY-M S.R.L.

Done Start International Invoice Microsoft Word - Document3 Local intranet zone 6:07 PM


FIGURE 4.5 HEADER INFORMATION

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://nismab2a/StateU/DEFAULT.HTM>



**Header Information**  
INTL INVOICE NUMBER IT WILL WORK : 9548

Update New

The record

No Records Available

International Invoicing

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

Done Start International Invoice Microsoft Word - Document3 Local intranet zone 6:12 PM

FIGURE 4.6 HEADER INFORMATION

International Invoicing - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://irislab2a/StateU/DEFAULT.HTM> Links



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Header Information

INTL INVOICE NUMBER IT WILL WORK : 9548

Status: Ready for new record

Invoice: 9548

Dealer: 008094

Our Ord. No.:

Currency Type: USD

Ship Toxy: DARWIN IRRIGATION S

41 STUART HWY

STUART PARK

NO TERRITORY

AUSTRALIA

DARWIN IRRIGATION S

Sold To :

Insert Cancel

Microsoft Word - Document4

Start International Invoice ...

Local intranet zone

6:21 PM

FIGURE 4.7 STATUS READY FOR NEW RECORD

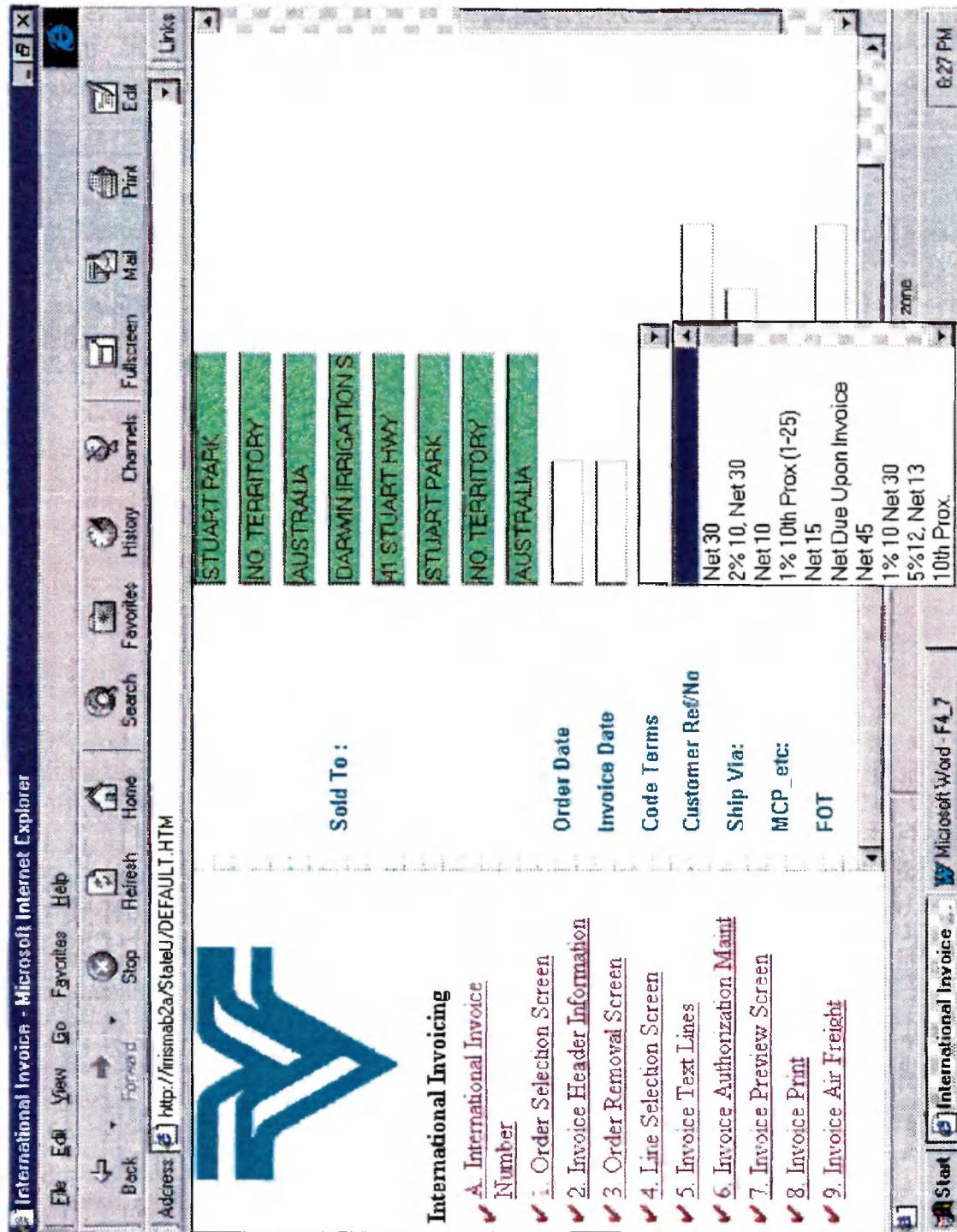



FIGURE 4.8 CODE TERMS IS A LIST BOX

International Invoicing - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address: <http://trismab2a/StateU/DEFAULT.HTM>



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

**Sold To :**

AUSTRALIA

DARWIN IRRIGATION S

41 STUART HWY

STUART PARK

NO. TERRITORY

AUSTRALIA

**Order Data**

**Invoice Date**

**Code Terms**

**Customer Ref/No**

**Ship Via:**

**MCP\_etc:**

**FOT**

**Employee Authorization**

**Ppd. or Coll or PP Add**

S. OSBORN, INVO CING COORDINATOR

LOLLY COOK, PROGRAMMER

JULIE ZAKOVEC, INVOICING COORDINATOR

JOHN MEAHL, INTERNATIONAL TRANSPORTATION MANA

MARY C. PARRISH, MANAGER, LOGISTICS

MIKE STATHES, PROGRAMMER

Michelangelo Tes: It is modified

Testing AT 8:59 AM

Test 12:53 pm

PAULA HAYNES, SALES ADMINISTRATOR

Local intranet zone

Microsoft Word - Document8

Start International Invoice ...

8:41 PM

FIGURE 4.9 EMPLOYEE AUTHORIZATION IS A LIST BOX

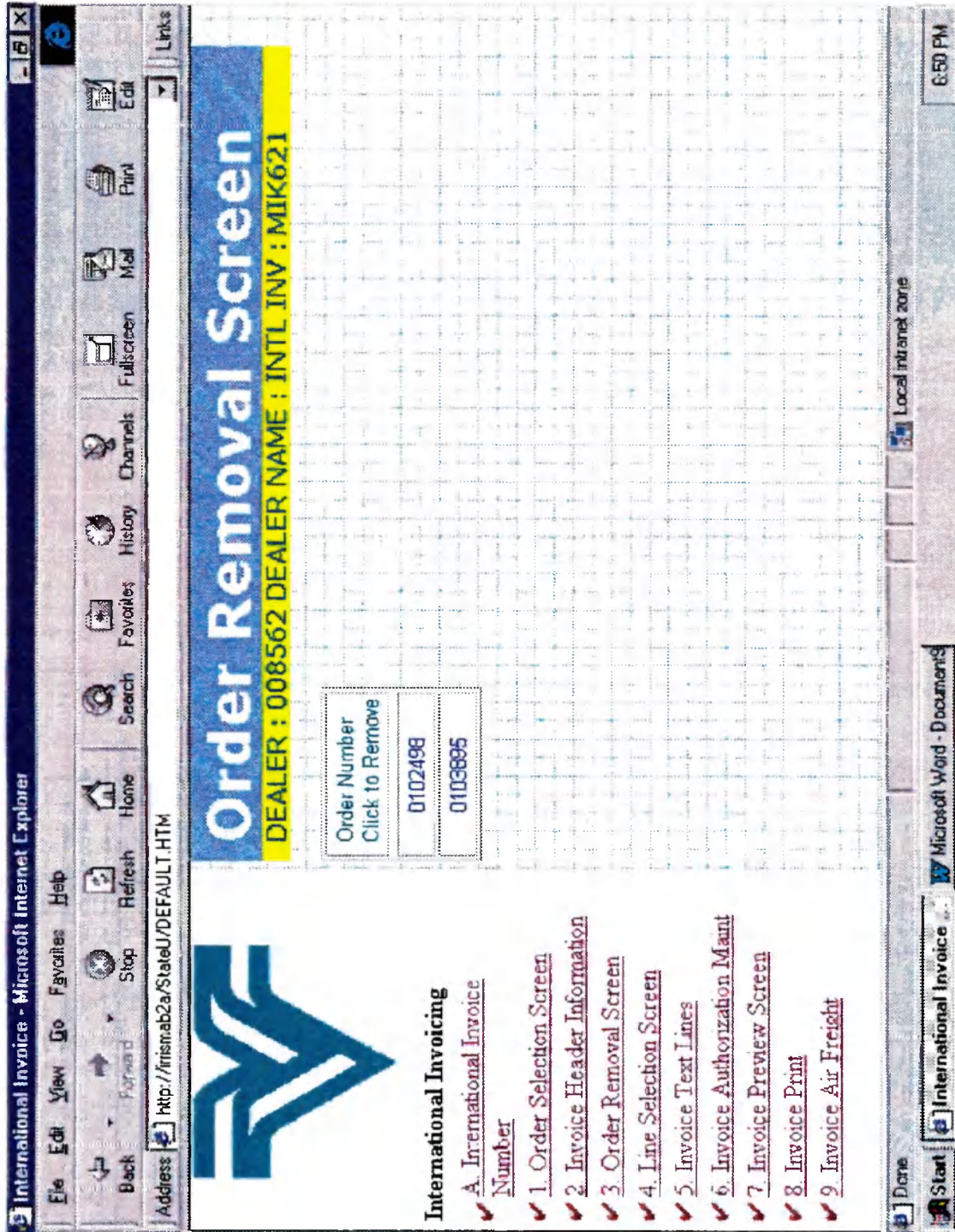



FIGURE 4.10 ORDER REMOVAL SCREEN

International Invoicing - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://nismab2a/StateU/DEFAULT.HTM>



**International Invoicing**

- [A. International Invoice Number](#)
- [1. Order Selection Screen](#)
- [2. Invoice Header Information](#)
- [3. Order Removal Screen](#)
- [4. Line Selection Screen](#)
- [5. Invoice Text Lines](#)
- [6. Invoice Authorization Maint](#)
- [7. Invoice Preview Screen](#)
- [8. Invoice Print](#)
- [9. Invoice Air Freight](#)

## Line Selection Screen

INTL INVOICE NUMBER MIKO\_GROW : MIK621

Line Choice	Qty Ordered	Qty Shipped	Qty Prevshp	Back Ordered	Part Number	Order Number	Line Price	Line Disc
00400	1	1	0	0	9360099	0102498	\$176.47	20.00
00600	3	3	0	0	K401244	0102498	\$1.93	20.00
00700	3	3	0	0	0232001	0102498	\$0.09	20.00
01000	6	6	0	0	0994507	0102498	\$2.29	20.00
01100	3	3	0	0	0994520	0102498	\$14.84	20.00

Start International Invoice - Micro... Microsoft Word - Document1

Local intranet zone

10:23 AM

FIGURE 4.11 LINE SELECTION SCREEN

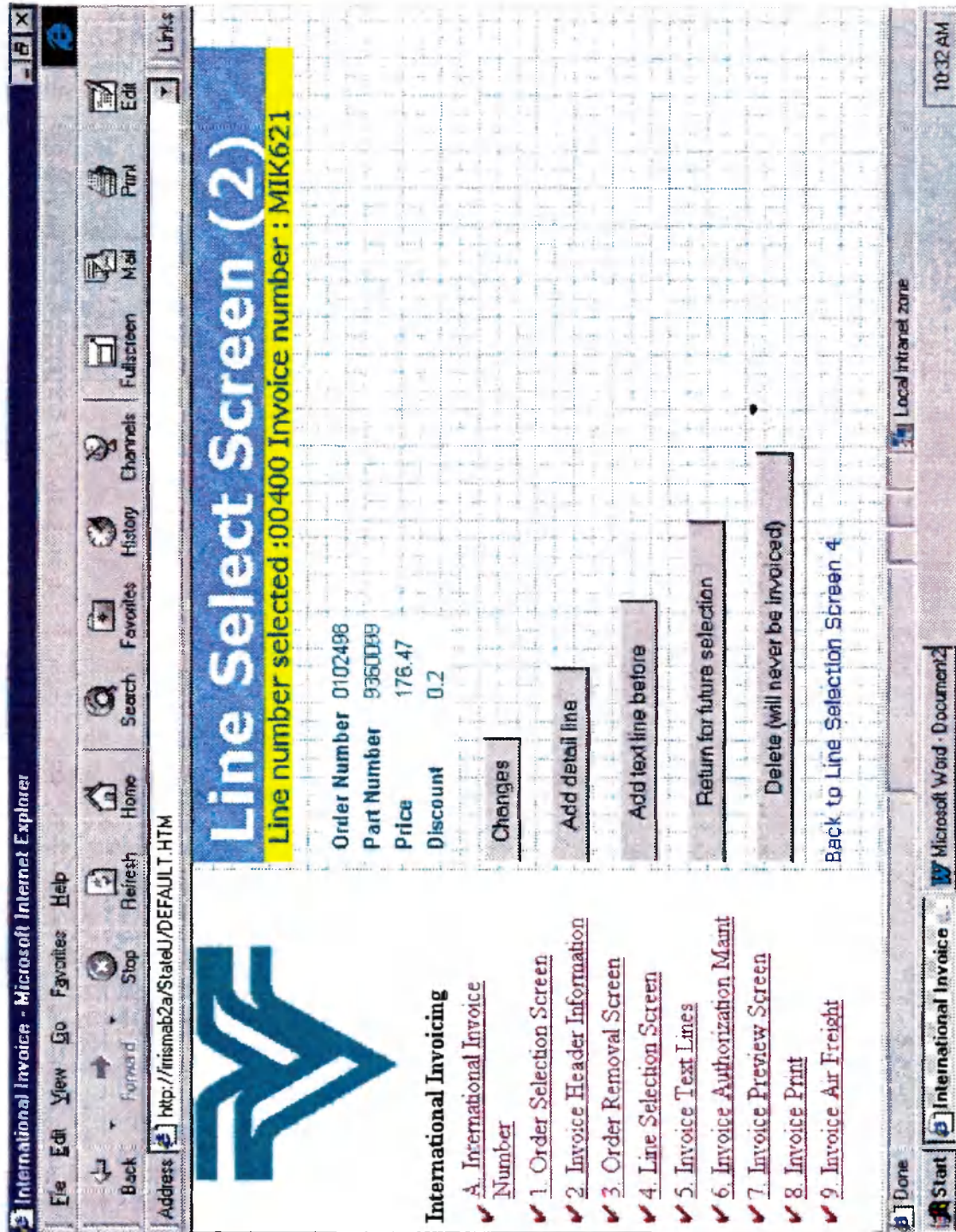



FIGURE 4.12 LINE SELECTED

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://irisrab2a/StateJ/DEFAULT.HTM>



**Option 4\_change**  
4 change

Please Make Your Changes

Part Number:	9360099
Price:	176.47
Discount:	0.2
Q shipped:	1
Description:	ASSM PIVOT PIPE 6.625 X 22-5.28 90 SPAC
Incl./N/C:	1

[Back to Invoice Text Lines Screen 5](#)

**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

Start International Invoice Microsoft Word F4.13 Local intranet zone 10:37 AM


FIGURE 4.13 OPTION4\_CHANGE

International Invoicing - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://irisrab2a/StateU/DEFAULT.HTM> Links



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Add a Detail Line

Invoice number: MIK621

Line Number:	
Qty Ordered:	
Qty Shipped:	
Part Number:	
Order Number:	
Line Discount:	
N/C_Incl_Flag:	
Unit Price:	
Description :	

[Back to Invoice Text Lines Screen 5](#)

Local intranet zone 10:45 AM

Microsoft Word - F4\_14

Start International Invoice

FIGURE 4.14 ADD A DETAIL LINE

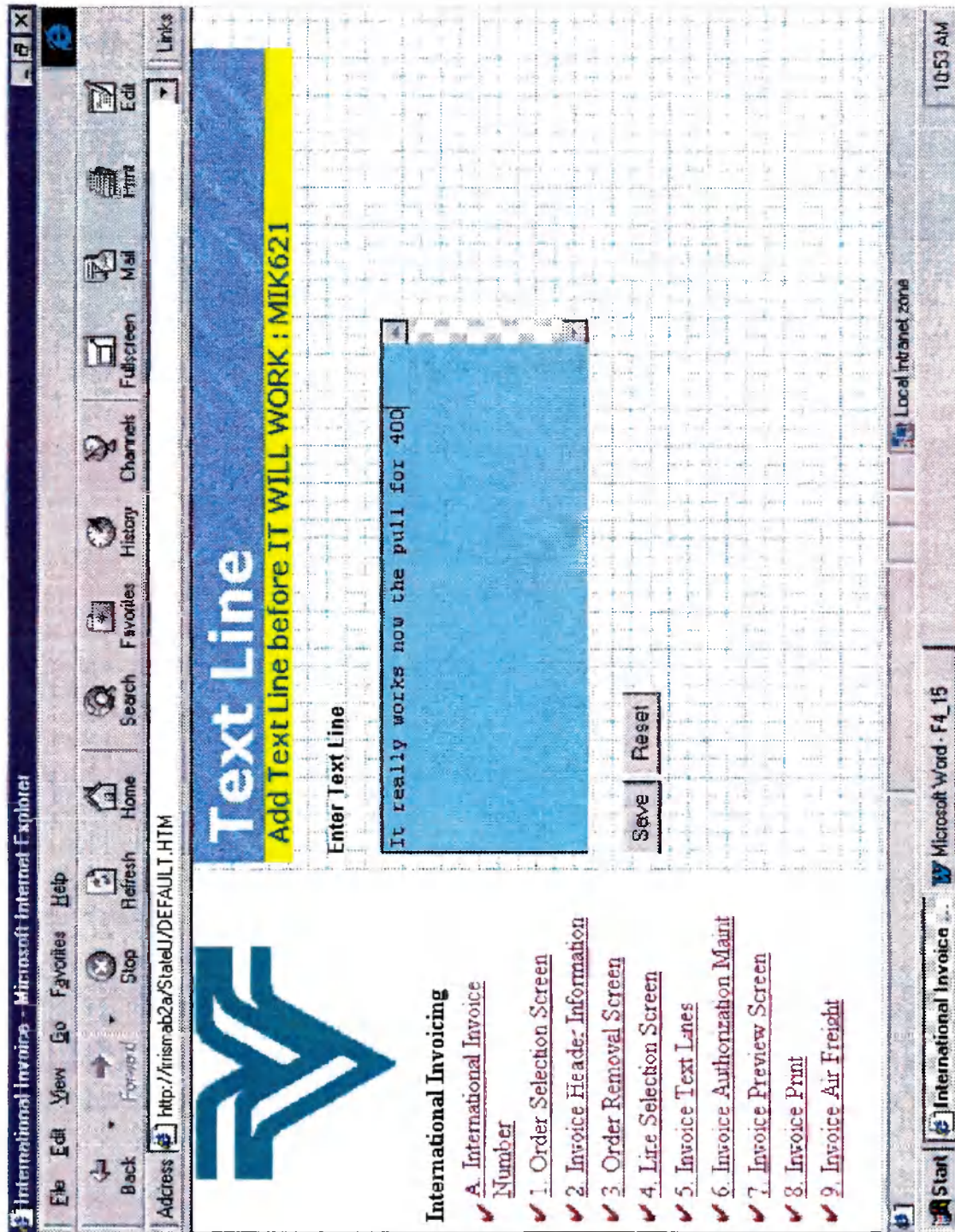


FIGURE 4.15 ADD A TEXT LINE



FIGURE 4.16 INVOICE TEXT LINES

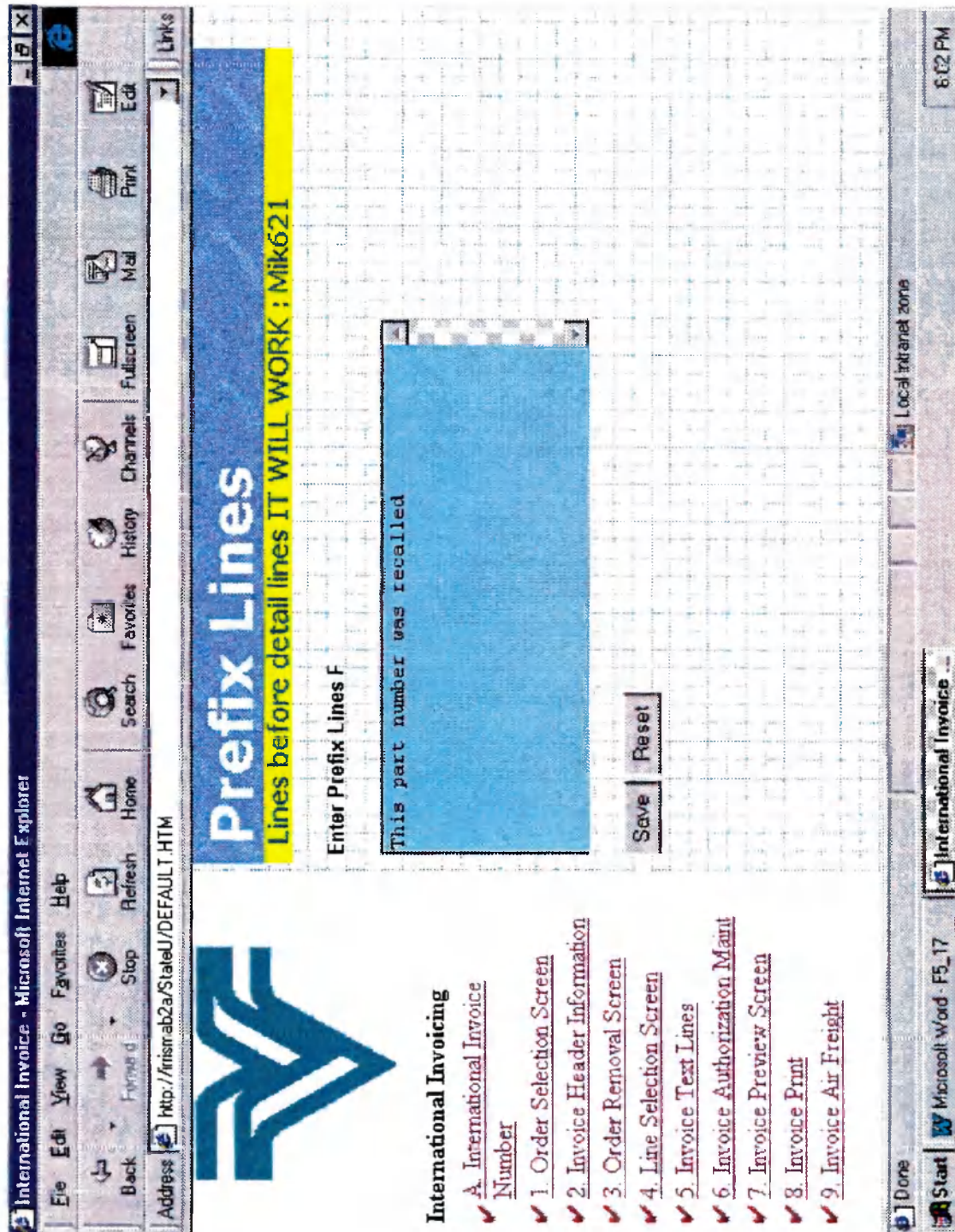



FIGURE 4.17 PREFIX LINES (LINES BEFORE DETAIL LINES)

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://nismab2a/StateU/DEFAULT.HTM> Links



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Text Total Lines

Success, Invoice:Mik621 Currency:USD

Sub Total	\$108.82	44444	64
Freight	700.5	55555	64
Extra Charges	0	55555	38844
Extra Charges	0	55555	51807
Extra Charges	0	55555	60871
Extra Charges	0	55555	83251
Grand Total	\$809.32	66666	64

Save Reset

Start Microsoft Word - F5\_17 International Invoice ... Local intranet zone 6:08 PM

FIGURE 4.18 TOTAL LINES

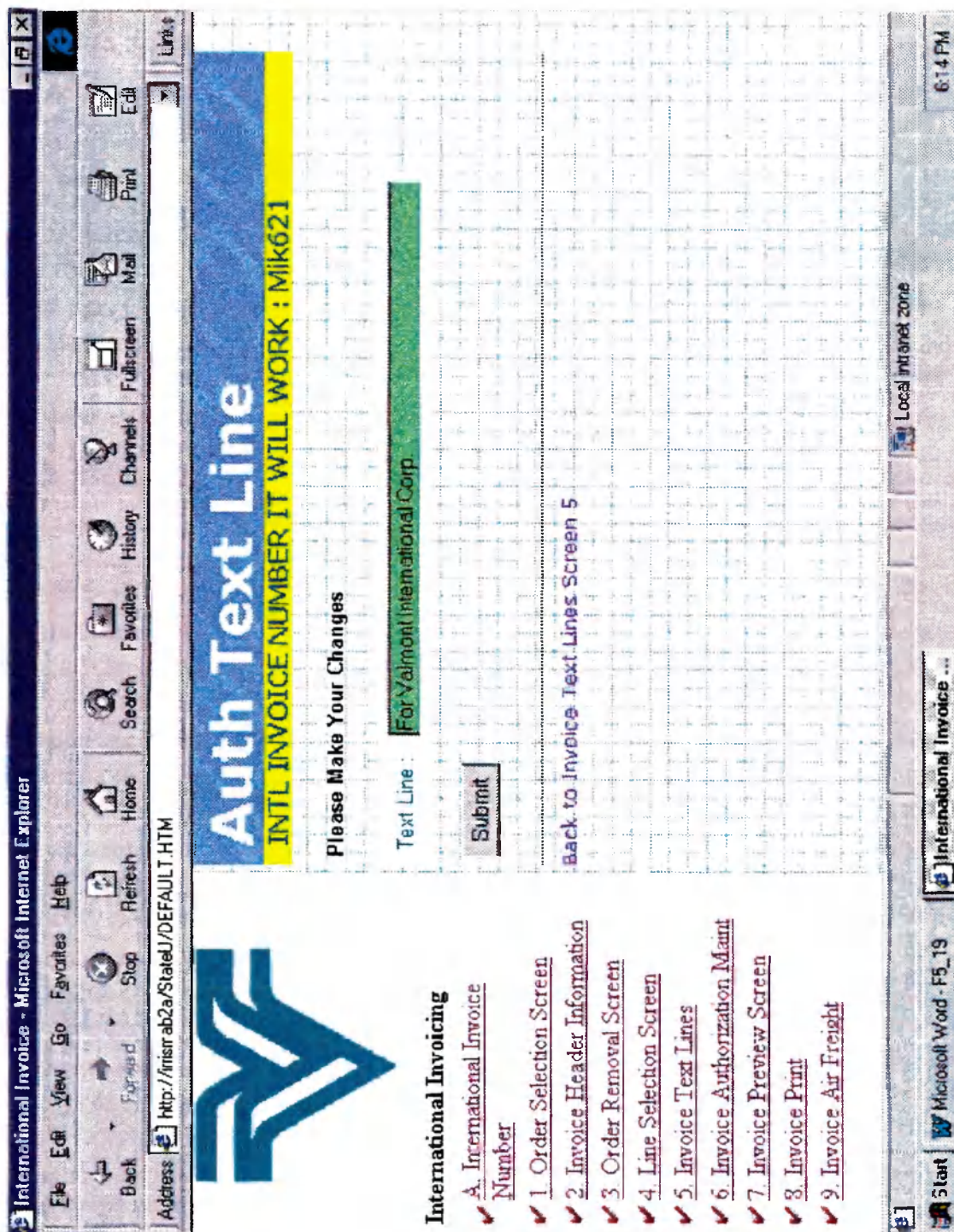



FIGURE 4.19 TO CHANGE (FOR VALMONT INDUSTRIES. INC)

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://nismab2a/StateU/DEFAULT.HTM>



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Maintenance Screen

For Selection click Employee ID

Employee ID	Name and Title
6015	S. OSBORN, INVOICING COORDINATOR
6204	LOLLY COOK, PROGRAMMER
6396	JULIE ZAKOVEC, INVOICING COORDINATOR
6671	JOHN MEAHL, INTERNATIONAL TRANSPORTATION MANAGER
6869	MARY C. PARRISH, MANAGER, LOGISTICS
6875	MIKE STATHES PROGRAMMER
9999	Michelangelo Test It is modified
9888	Testing AT 8:59 AM
7654	Test 12:53 pm
3809	PAUL A. HAYNES/SALES ADMINISTRATOR

Start Microsoft Word - F5\_19 International Invoice ... Local intranet zone 6:18 PM

FIGURE 4.20 MAINTENANCE SCREEN

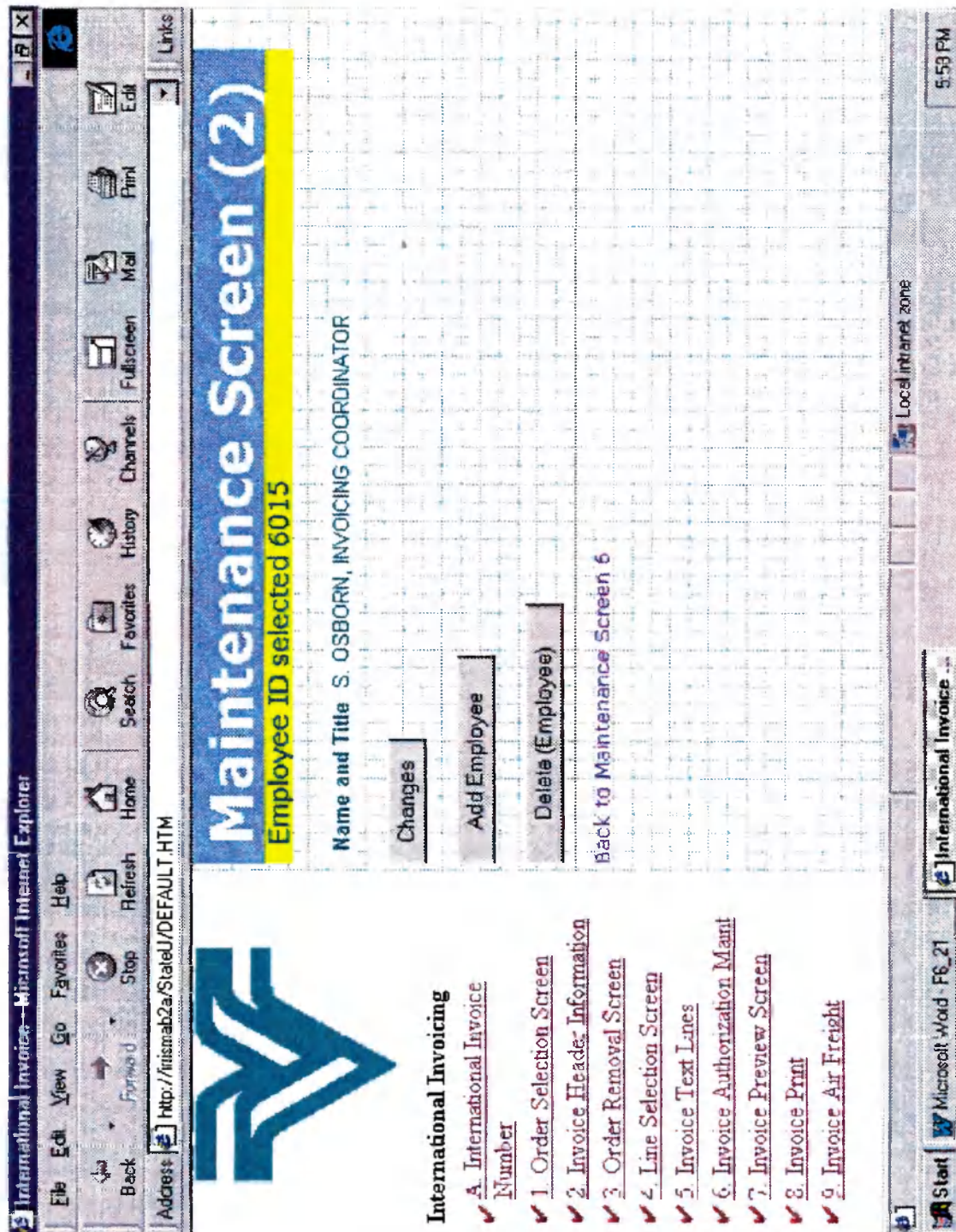


FIGURE 4.21 MAINTENANCE SCREEN (2)

International Invoice - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit

Address <http://irisab2a/StateU/DEFAULT.HTM>



**International Invoicing**

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

## Preview Screen A1

MIKO\_GROW INVOICE:MIK621 DEALER :IRRICAMPO

**Ship To** VALLEY-M S.R.L  
123456789012345678901234567890  
AD\_LINE2  
CHISINAU  
REP. OF MOSTRATAN

**Sold To** VALLEY-M S.R.L  
STR. A RUSSO 1,CA 206  
AD\_LINE2  
CHISINAU  
REP. OF MOSTRATAN

**Terms:** Sight Draft

Our Order No	Order Dta	Invoice No	Date	Customer's order or reference	Currency

6:02 PM

Start Microsoft Word - FS\_21 International Invoice ... Local intranet zone

FIGURE 4.22 PREVIEW SCREEN START (A)

International Invoicing

- ✓ [A. International Invoice Number](#)
- ✓ [1. Order Selection Screen](#)
- ✓ [2. Invoice Header Information](#)
- ✓ [3. Order Removal Screen](#)
- ✓ [4. Line Selection Screen](#)
- ✓ [5. Invoice Text Lines](#)
- ✓ [6. Invoice Authorization Maint](#)
- ✓ [7. Invoice Preview Screen](#)
- ✓ [8. Invoice Print](#)
- ✓ [9. Invoice Air Freight](#)

Address: <http://irisnab2a/StateJ/DEFAULT.HTM>

Our Order No:   
 Invoice No: milk621   
 Date:   
 Customer's order or reference:   
 Currency: USD

Ppd. Coll. P.P. & Add Ship Via: F.O.B. Point Sales Tax   
 AIR FREIGHT ----- MCPETC For Point EXEMPT

This part number was recalled

Item No	Quantity Ordered	Quantity This Shipment	Quantity Prev Shipped	Qty Remain On Order	Part No	Description	Unit Price	Total Amount
00400	1	1	0	0	9360099	ASSM PIVOT PIPE 8.625 X 22-5.28 90 SPAC	Includ	

Start Microsoft Word - F6\_23 International Invoicing -- Local intranet zone 6:09 PM

FIGURE 4.23 PREVIEW SCREEN CONTINUE (B)

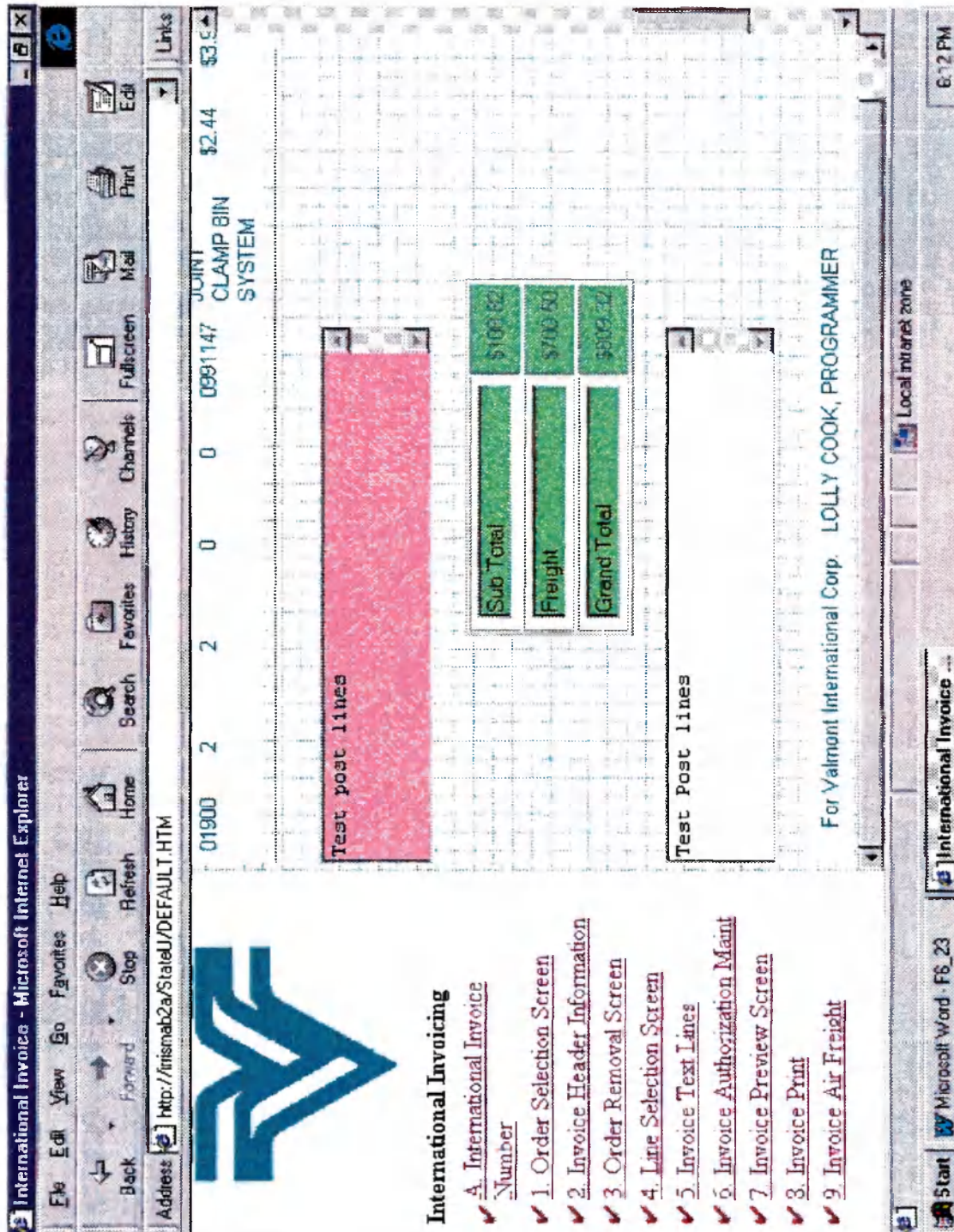


FIGURE 4.24 PREVIEW SCREEN END (C)

# INVOICE

SHIP TO ALKHORAYEF IRR SYSTEM CO

P.O. BOX 42852

RIYADH

SAUDI ARABIA

SOLD TO

00008065

ALKHORAYEF IRR SYSTEM CO

P.O. BOX 42852

RIYADH

SAUDI ARABIA

VALMONT INTERNATIONAL CORP.  
San Antonio, Texas  
A Subsidiary of  
VALMONT INDUSTRIES, INC.

Phone (402) 359-7201 Duns No. VJ-956-6036  
Fax (402) 359-4429 E-Mail vintlevmont.com  
Valley, Nebraska 68864 USA

Please refer to our invoice number when making your remittance.

Please mail remittance to: Valley, Nebraska 68064 USA

Items not shipped have been back ordered and will follow as soon as possible  
(A separate invoice will be issued.)

We will not accept returned goods except on our written authorization.

Returned goods must be accompanied by packing slip.

Valmont has complied with the provisions of the Fair Labor Standards Act of  
1938 as amended in producing this merchandise.



TERMS: Sight Draft

OUR ORDER NO	ORDER DTE	INVOICE NO	DATE	CUSTOMER'S ORDER OR REFERENCE NO	CURRENCY
SEE BELOW	7/22/99	7100099	8/27/99	15538,605,37990,15780,38176	USD
PPO.	COLL	P.P. & AOO	SHIP VIA	F.O.B. POINT	SALES TAX
			OCEAN FREIGHT-SJF187	VALLEY NEBRASKA USA	EXEMPT

ITEM NO	QUANTITY ORDERED	QUANTITY THIS SHIPMENT	QTY PREV SHIPPED	QTY REMAIN ON ORDER NO	DESCRIPTION	UNIT PRICE	TOTAL AMOUNT
075	460	460	0	0	0232004 VLV MS LOWER DRAIN BODY	\$1.01	\$464.60
076	460	460	0	0	0246170 FIT MS UPPER DRAIN BODY	\$0.41	\$188.60
077	2000	2000	0	0	0271013 HSE CL 1-1/16 TO 2.00	\$0.25	\$520.00
078	7200	7200	0	0	0271021 IRR MS CLAMP-HYD TUBE-24 IN F/IN PIPE 43	\$0.24	\$1,728.00
079	2700	2700	0	0	0271024 IRR MS CLAMP-PIPE LINE 8 IN SYSTEM	\$0.25	\$675.00
080	16200	16200	0	0	0271040 HSE MS CABLE CLAMP 42	\$0.10	\$1,620.00
081	1600	1600	0	0	0271082 HSE IT 3/4I NPT FEMALE X 3/4I HOSE BARB	\$0.40	\$640.00
082	9500	9500	0	0	0271084 HSE CL 1 1/4I S.S. HOSE DROP CLAMP	\$0.15	\$1,425.00
083	100	100	0	0	0271087 IRR MS 6 5/8 SCREW CLAMP STAINLESS STEEL	\$0.6	\$61.00
084	69	69	0	0	0272033 HSE WT 3/4I FLEX X 250 F T. ROLL	\$31.25	\$2,156.25
085	256	256	0	0	03E1558 ELC MR (GRN) 68 RPM US M OTORG.BOX ASSM	\$216.86	\$55,521.28
086	100	100	0	0	0314343 ELC FS MAIN PANEL 12 AMP 600V	\$1.75	\$175.00
087	1200	1200	0	0	0314466 ELC MS STRAINRELIEFHUB11 NF/SHIELDED CABLE	\$1.49	\$1,788.00
088	10000	10000	0	0	0314571 ELC WR POWER CABLE-4 CON DUCTOR	\$0.30	\$3,000.00

Invoice #100099 continue

## INVOICE

VALMONT INTERNATIONAL CORP.  
San Antonio, Texas  
A Subsidiary of  
VALMONT INDUSTRIES, INC.

SHIP TO ALKHORAYEF IRR. SYSTEM CO

P.O. BOX 42352

Phone (402) 359-2201 Duns No. 09-356-6036  
Fax (402) 359-4429 E-Mail vint@valmont.com  
Valley, Nebraska 68064 USA

RIYADH

SAUDI ARABIA

SCLD TO

00008065

ALKHORAYEF IRR. SYSTEM CO

P.O. BOX 42852

Please refer to our invoice number when making your remittance.  
Please mail remittance to: Valley, Nebraska 68064 USA

Items not shipped have been back ordered and will follow as soon as possible  
(A separate invoice will be issued.)

We will not accept returned goods except on our written authorization.  
Returned goods must be accompanied by packing slip.

RIYADH

SAUDI ARABIA

Valmont has complied with the provisions of the Fair Labor Standards Act of  
1938 as amended in producing this merchandise.



## TERMS: Sight Draft

OUR ORDER NO	ORDER DTE	INVOICE NO	DATE	CUSTOMER'S ORDER OR REFERENCE NO	CURRENCY
SEE BELOW	7/22/99	7100099	8/27/99	15538.605.37950.15780.38176	USD
PPD.	COLL	P.P. & ADD	SHIP VIA	F.O.B. POINT	SALES TAX
			OCEAN FREIGHT--SJF187	VALLEY NEBRASKA USA	EXEMPT

TOTAL FOB VALLEY NEBRASKA

\$517,915.85

FREIGHT &amp; DOCUMENTS

\$44,700.00

TOTAL C &amp; F JEDDAH

\$562,615.85

For Valmont International Corp. MARY C. PARRISH, MANAGER, LOGISTICS

# EXTRACTED FROM LINKS\_HTM

```

<html><head>
<table border="0">
<tr>
<th align="left" colspan="2" bgcolor="#FFFFFF">International
Invoicing</th>
<th align="left" colspan="2" bgcolor="#FFFFFF">&nbsp;&nbsp;&nbsp;</th>
</tr>
<tr>
<td></td>
<td><a href="mailto:stateU/profile.asp">A. International
Invoice Number</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:stateU/transcript.asp">1. Order Selection
Screen</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:stateU/Rau_55HTM.asp">5. Invoice Text
Lines</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:stateU/MENU_6.asp">6. Invoice
Authorization Maint</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:stateU/Raul_77.asp">7. Invoice
Preview Screen</a></td>
</tr>
</table>
</body>
</html>

```

EXTRACT X0 Links\_htm Related to Fig. 4.1

## EXTRACTED FROM PROFILE\_ASP

```

<% Language=VBScript %>
<% If Not (IsEmpty(Request.Form("txtClassID"))) Then
    Dealer = Request.Form("txtClassID")
    Session("dealer") = dealer
    If Instr(Session("requestedPage"), "profile") = 0 Then
        Response.Redirect Session("requestedPage")
    Else
        Response.Redirect "miko_home.htm"
    End If
End If %>
<HTML><HEAD>
<script ID="clientEventHandlersVBS" LANGUAGE="vbscript">
Sub cboClass_OnClick
    intIndx = frmClsVw.cboClass.selectedIndex
    strClass = frmClsVw.cboClass.options(intIndx).text
    strClass = Trim(strClass)
    strTemp = strClass
    intPos1 = Instr(strTemp, ",")
    frmClsVw.txtClassID.value = Left(strTemp, intPos1-1)
End Sub
</script>
</HEAD><BODY>
<H1>Invoice and Dealer</H1>
<% Set zsql = Server.CreateObject("Enroll.Enrollment")
    VCUST1 = zsql.VCUSTall()
    x0 = zsql.MIKO_IA_PULL() %>
<% if VCUST1(0,0) > 0 then %>
    <form action="profile.asp" method="post" id="frmClsVw" name="frmClsVw">
    <select id="cboClass" name="cboClass" size="2" style="HEIGHT: 120px; WIDTH: 82px">
    <% if 1 = 1
        do while 11 < VCUST1(0,0) + 1
            Response.Write "<OPTION>" & VCUST1(0,11) & "," & x0 & "2,1,USD," & VCUST1(1,11)
            11 = 11 + 1
        loop %>
    </select>
    </form>
    <% end if %>
</BODY></HTML>

```

EXTRACT X1 Profile.asp Related to Fig 4.2

**EXTRACTED FROM GLOBAL\_ASA**

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnStart
    profilePage = "/StateU/profile.asp"
    currentPage = Request.ServerVariables("SCRIPT_NAME")
    Session("requestedPage") = currentPage
    If StrComp(currentPage,profilePage,1) Then
        Response.Redirect(profilePage)
    End If
End Sub
</SCRIPT>
```

**EXTRACTED FROM PROFILE\_ASP**

```
<% Language=VBScript %>
<%
    If Instr(Session("requestedPage"), "profile") = 0 Then
        Response.Redirect Session("requestedPage")
    Else
        Response.Redirect "miko_home.htm"
    End If
%>
```

**EXTRACT X2 Page Redirection, Related to Fig 4.2**

**EXTRACTED FROM VCUSTall**

```

Public Function VCUSTall()
    Dim conn As ADODB.Connection
    Dim rs As Object
    Dim vROBA() As Variant

    strSQL = " SELECT U#CM_ADDR, COMPANY_NAME FROM DEALERS_VIEW " & _
        " where U#CM_ADDR > 'ALFAL' ORDER BY U#CM_ADDR "

    Set conn = CreateObject("ADODB.connection")
    conn.Open "Data Source=IX; USER ID=IX; PASSWORD=IX"
    Set rs = conn.Execute(strSQL)

    Do While Not rs.EOF
        iRC = iRC + 1
        For iFC = 0 To rs.Fields.Count - 1
            vRDBA(iFC, iRC) = rs(iFC).Value
        Next
        rs.MoveNext
    Loop

    VCUSTall = vRDBA

    conn.Close

End Function

```

**EXTRACTED FROM PROFILE\_ASP**

```

< & Set zsoel = Server.CreateObject("Enroll.Enrollment")
    VCUST1 = zsoel.VCUSTall()
    x0 = zsoel.MIKO_IJA_FULL() &>

```

**EXTRACT X3 Component Calling, Related to Fig 4.2**

## EXTRACTED FROM TRANSCRIPT\_ASP

```

<% Language=VBScript %>
<HTML><BODY>
<H1>Order Selection Screen</H1>
<% strCusNum = Session("dealer")
   Set EnrollObj = Server.CreateObject("Enroll.Enrollment")
   VRDBA = EnrollObj.RIOR_SEL VB(strCusNum) %>
<TABLE border=1 cellpadding=3 cellspacing=3>
  <TR><TD>Order Number<BR>Click to Select</TD>
  <TD>Number of lines</TD></TR>
  <TR bgcolor=white>
    <tdo while ll < 7
      stext = stext & "<TR align=center><TD>"
      stext = stext & "<A HREF='bid.asp?item=" & VRDBA(0,ll) & "'>"
      stext = stext & VRDBA(0,ll) & "</a>" & "</TD><TD>"
      stext = stext & VRDBA(1,ll) & "</TD>" & "</TR>"
      ll = ll + 1
    loop
    Response.Write stext
    VRDBA(1,0)= ll <TR>
  <TD> <form action="next.asp" id=form1 name=form1>
    <input type="submit" name="next" value="next"></form> </TD></TR>
</Table>
</BODY></HTML>

```

## EXTRACT X4 Transcript.asp, Related to Fig 4.3



**EXTRACTED FROM VBLINE\_1**

```

Public Function VBlne_1(ByVal line_num As String, ByVal _
    cus_num As String, ByVal sor_num As String, _
    ByVal inv_num As String) As Integer
    On Error GoTo ErrorHandler
    Dim conn As ADODB.Connection
    Dim strUpdateIR, strUpdateIID As String, available As Integer

    Set conn = CreateObject("ADODB.Connection")
    conn.Open "Data Source=IX; USER ID=IX; PASSWORD=IX"
    available = FindAvailable(conn, line_num, cus_num, sor_num)
    strUpdateIRSQL = "Update IIR Set IIR.SQRL_QUANTITY_II_SHIPPED ..."
    strUpdateIIDSQL = "Update IID Set IID.SQRL_QUANTITY_SHIPPED ..."

    conn.BeginTrans
    conn.Execute strUpdateIRSQL
    conn.Execute strUpdateIIDSQL
    conn.CommitTrans

    conn.Close
Exit Function

ErrorHandler:
    If Not IsEmpty(conn) Then conn.RollbackTrans
    VBlne_1 = 2
End Function

```

**EXTRACT X6 ACID properties, Related to Fig 4.4**

## EXTRACTED FROM CLASS77FORM\_ASP

```

<OBJECT ID="rsRau9555A1SQLQuery" WIDTH=151 HEIGHT=24
CLASSID="CLSID:F602E721-A281-11CF-A5B7-0080C73AAC7E">
  <PARAM Name="RangeType" Value="1">
  <PARAM Name="DataConnection" Value="StatelU">
  <PARAM Name="CommandType" Value="0">
  <PARAM Name="CommandText" Value="SELECT IIH.IIHR_INTL_INVOICE_NUMBER, " & -
    " IIH.CUSTOMER_NUMBER,
    " IIH.INVC_ORD_NUM,
    " IIH.II_CJRR_TYP,
    " IIH.IIHR_SHIP_ADDRESS_LINE1,
    " IIH.IIHR_SHIP_ADDRESS_LINES,
    " IIHR_SOLD_ADDRESS_LINE1,
    " IIHR_SOLD_ADDRESS_LINES,
    " II_DATE,
    " INVCAT_ORD,
    " IIHR_CODE_TERMS,
    " IIHRM_DOD_TXT,
    " IIHR_CUSTOMER_ORD_REF_NUMBER,
    " IIHR_SHIP_TEXT,
    " II_MAST_INTL,
    " IIHR_FOT_POINT,
    " EMPLOYEE_NUMBER,
    " II_INVC_PRT_FLG
  FROM IIH WHERE IIH.IIHR_INTL_INVOICE_NUMBER = '" & -
  Session("major") & "' ">

```

&lt;/OBJECT&gt;

EXTRACT X7 Related to Fig 4.5

## EXTRACTED FROM CLASS77ACTION\_ASP

```

<#
  inv_num = rsRau9553AISQLQuery("IHR_INTL_INVOICE_NUMBER")
  varRec = Sesion("raul77")
  strInIHSQL = "INSERT INTO IIH ("
  For i = 1 To (varRec(0, 0) - 1)
    strInIHSQL = strInIHSQL & varRec(i, 0) & ", "
  Next
  strInIHSQL = strInIHSQL & varRec(varRec(0, 0), 0) & ") VALUES("
  For i = 1 To (varRec(0, 0) - 1)
    strInIHSQL = strInIHSQL & varRec(i, 1) & ", "
  Next
  strInIHSQL = strInIHSQL & varRec(varRec(0, 0), 1) & ")"
  Set x = Server.CreateObject("Enroll.Enrollment")
  x77 = x.Veline_SOS_B(strInIHSQL)

  If x77 = 0 then
    x77 = x.MIKO_IJA_UPD()
  end if >

```

## EXTRACTED FROM VELINE\_SOS\_B

```

Public Function Veline_SOS_B(ByVal strUpIHSQL As String) As Integer

  Set rs = conn.Execute(strUpIHSQL)
  conn.Close
  Set conn = Nothing

End Function

```

EXTRACT X8 Related to Fig 4.7

## EXTRACTED FROM MARIA\_3\_ASP

```

<% Language=VBScript %>
<% frmStudentID = Session("id") %>
</HTML>
</HEAD>
</HEAD>
<BODY>
<H1>Order Removal Screen</H1>
<H3>
    DEALER :   <%Session("dealer")%>
</H3>
<%
    strInvNum = Session("major")
    Set EnrollObj = Server.CreateObject("Enroll.Enrollment")
    VRDBA = EnrollObj.vCallStoredProc(strInvNum) %>
<TABLE border=1 cellpadding=3 CELLSPACING=3>
<TR bgcolor=silver><B><TD>Order Number<BR><Click to Remove</TD></B></TR>
<TR bgcolor=white>
<%
    size1 = VRDBA(0,0) + 1
    do while i1 < size1
        strText = ""
        Response.Write "<TR align=center><TD>"
        Response.Write "<A href='\"Bld3_new.asp?Item= \" & VRDBA(0,1) & \"'>"
        Response.Write VRDBA(0,1)
        strText = strText & "</a>"
        strText = strText & "</TD>"
        strText = strText & "</TR>"
        Response.Write strText
        i1 = i1 + 1
    loop %>
</Table>
</BODY>
</HTML>

```

EXTRACT X9 Maria\_3.asp Related to Fig 4.10

#### EXTRACTED FROM ACTIVE SERVER PAGE BID3\_NEW\_ASP

```
<% Set EnrollObj = Server.CreateObject("Enroll.Enrollment") %>
errEnrollment = EnrollObj.IID_IIR3(strInvNum, strSorNum, strCusNum) %>
```

#### EXTRACTED FROM COMPONENT IID\_IIR3

```
Public Function IID_IIR3(ByVal inv_num As String, ByVal sor_num As String, ByVal cus_num As String)
    cmd1.CommandType = adCmdStoredProc
    cmd1.CommandText = "IIR3_1A"
    cmd1.ActiveConnection = conn
    Set rs1 = cmd1.Execute!, Array(vrDBA(1, 1), cus_num, sor_num, vrDBA(0, 1), inv_num))
End Function
```

#### EXTRACTED FROM STORED PROCEDURE IIR3\_1A

```
CREATE PROCEDURE IIR3_1A
(@quantity int, @cus_num varchar(6), @sor_num varchar(7),
@line_num varchar(5), @inv_num varchar(6) ) AS
BEGIN TRAN
    UPDATE IIR
        SET SORL_QUANTITY_II_SHIPPED = SORL_QUANTITY_II_SHIPPED - @quantity
        WHERE IIR.CUST_NUMBER = @cus_num AND IIR.SORD_NUMBER_1_7 = @sor_num
    DELETE FROM IID
        WHERE IIDR_INTL_INVOICE_NUMBER = @inv_num AND SORD_NUMBER_1_7 = @sor_num
    if @@error > 0
        begin
            ROLLBACK TRAN
            return 99999
        end
    else
        begin
            COMMIT TRAN
            return 0
        end
end
GO
```

EXTRACT X10 ACID Properties, Using a Stored Procedure

**EXTRACTED FROM BID4\_ASP**

```

<HTML><HEAD>
<script language="vbscript">
Sub Add_Click()
    button = "Add"
    window.navigate "Raul78_4.asp?LineNum=" & lineNumber.value & _
        "&sortNum7=" & sortNumber.value & _
        "&button=" & button
End Sub
</script>
</HEAD><BODY>
<H1>Line Select Screen (2)</H1>
<table>
  <tr> <td> <B>Order Number      </B> </td><td><!--Request("Item1")%>  </td> </tr>
</table>

<INPUT TYPE=BUTTON VALUE="Add detail line" ONCLICK="Add_Click" NAME="Add">
<br>
<a href="Raul78_4.asp">Back to Line Selection Screen 4</a>
</BODY></HTML>

```

**EXTRACT X11. Bid4.asp Related to Fig 4.12**

## EXTRACTED FROM RAU4\_CHANGES\_ASP

```

<% Language=VBScript %>
<If Not (IsEmpty(Request.Form("rl")))) Then
    Set x = Server.CreateObject("Enroll.Enrollment")
    parNumber = request("rl")
    x0 = x.VB_4_31(strInNum, strOutNum, strLinNum, parNumber, _
        price, discount, q_ship, descri, i_no)
    Response.Redirect "Raul78_4.asp"
End If%>

<% strLinNum = request("LinNum") %>
<HTML><BODY>
<H1>Option 4_Change</H1>
<p><b>Please Make Your Changes </b></p>
<form action="Raul4_Changes.asp" method="post" name="ft1" id="ft1">
<TABLE border=1 cellpadding="3">
<% VRDBA = Session("rs_4ch")
    size1 = 7 %>
<% do while !< size1 %>
<% raul = "r" & CStr(i1) %>
<tr><td>
    <font face="Arial, Helvetica, sans-serif"><%=varName(i1)%> </font></td>
<td><input type="text" style="BACKGROUND-COLOR: lightgreen"
    name="<%=response.write raul%>" value="<%=Select Case i1
        Case 1
            response.write parNumber
        End Select %>
        " size="40"></td></tr>
<% i1 = i1 + 1
loop%>
</Table>
<p><input type="submit" name="B1" value="Submit"></p>
</form>
<a href="Rau_55HTM.asp">Back to Invoice Text Lines Screen 5</a>
</BODY></HTML>

```

EXTRACT X12. Code Related to Fig 4.13

## EXTRACTED FROM NEW\_RAU44A1\_ASP

```

<% LANGUAGE=VBScript %>
<% frmFBk = Request.Form("taFeedBack"); %>
<% If Not IsEmpty(frmFBk) Then
    Set x = Server.CreateObject("Enroll.Enrollment")
    xl = x.VB_41_4txtUP(strInNum, Miko_sor, frmFBk, miko_tell, Miko_lin)
    Response.Redirect "Raul78_4.asp"
Else %>
</html></head>
<script language="VBScript">
Sub btnSubmit1_OnClick ()
    frmFB.Submit
End Sub
</script>
</head><body>
<h1>Text Line</h1>
<% strInNum = request("LinNum")
<%Set EnrollObj = Server.CreateObject("Enroll.Enrollment")
VRDBA = EnrollObj.VB_41_4txt(strInNum, strSorNum, strLinNum)%>
<form action="NEW_RAU44A1.asp" method="post" name="frmFB" id="frmFB">
<input TYPE="HIDDEN" NAME="Miko_tell" Value="<%
    IF VRDBA(0,0) = 0 THEN
        response.write ">"><%END IF%>">
<p><strong>Enter Text Line</strong></p>
<p><textarea name="taFeedBack"
    style="BACKGROUND-COLOR: lightblue; HEIGHT: 133px;
    WIDTH: 351px"><%IF VRDBA(0,0) > 0 THEN Response.Write VRDBA(1,0) END IF%>
</textarea></p>
<input type="button" id="btnSubmit1" name="btnSubmit1" value="Save">
<input type="reset" id="reset1" name="reset1" value="Reset">
</form>
</body></html><% End If %>

```

EXTRACT X13. Code Related to Fig 4.15

## EXTRACTED FROM ADDEFOUR\_ASP

```

<! Language=VBScript %>
<% strLinNum = request("LinNum") %>
<HTML><BODY>
<H1>Screen Confirmation (4)</H1>
<%
Set EnrollObj = Server.CreateObject("Enroll.Enrollment")
x_back = 3
Select Case button
Case "Delete"
    x_back = EnrollObj.DR4DELVB(strInvNum, strSorNum, strLinNum)
Case "Return"
    x_back = EnrollObj.DR4RETVB(strInvNum, strSorNum, strLinNum)
End Select
If x_back = 1 Then
    response.write "1 Error " & strSorNum
    response.write "1 Error " & messageR
    response.end
End If
If x_back = 0 Then
    response.write "4 OK MICHELANGELO, SUCCESS " & strSorNum
    response.write "4 OK MICHELANGELO, SUCCESS " & messageR
    response.end
End If
%>
</BODY></HTML>

```

EXTRACT XI4. Code Related to Fig 4.12 (Return, Delete, Options)

## EXTRACTED FROM RAU\_55HTM\_ASP

```

<!-- Language=VBScript -->
<HTML><BODY>

<H1>Invoice Text Lines</H1>
<p> <b> Please Click to Select </b> </p>

<table border="0">
  <tr>
    <td></td>
    <td><a href=" ../stateU/NEW_RAU55Al.asp">- Prefix Lines (Lines Before Detail Lines)</a></td>
  </tr>
  <tr>
    <td></td>
    <td><a href=" ../stateU/NEW_RAU55B1.asp">- Total Lines</a></td>
  </tr>
  <tr>
    <td></td>
    <td><a href=" ../stateU/NEW_RAU55C1.asp">- Post Lines (Lines After Detail Before Totals)</a></td>
  </tr>
  <tr>
    <td></td>
    <td><a href=" ../stateU/NEW_RAU55D1.asp">- Post Lines (Lines After Totals Before Auth)</a></td>
  </tr>
  <tr>
    <td></td>
    <td><a href=" ../stateU/Rau_55E.asp">- To Change (For Valmont Industries, Inc.)</a></td>
  </tr>
  <tr>
    <td></td>
    <td><a href=" ../stateU/Rau_55F1List.asp">- To Change Subtotal Verbage</a></td>
  </tr>
</body></html>

```

EXTRACT X15. Rau\_55htm.asp, Code Related to Fig 4.16

## EXTRACTED FROM NEW\_RAU55B1\_ASP

```

<!--Text Total Lines-->
<% Set EnrollObj = Server.CreateObject("Enroll.Enrollment")
VRDBA = EnrollObj.VB_41_52(strInNum)
Session("rau55b1") = VRDBA &
<form action="NEW_RAU55B1.asp" method="post" name="frmFBB" id="frmFBB">
<TABLE border=1 cellpadding="3">
<% il = 1
size1 = VRDBA(0,0) + 1 &
<do while il < size1 &
<% if il = 1 or il = VRDBA(0,0) then &
<% raul = "A" & CStr(il) &
<tr> <td>
<input type="Text" style="BACKGROUND-COLOR: lightgreen"
name="<response.write raul>" value="<response.write VRDBA(3,il)&" size="40">
</td> <td bgcolor="FFFFFF">
<response.write FormatCurrency(VRDBA(2,il))&
</td></tr>
<% else &
<% j = 2 &
<do while j < VRDBA(0,0) &
<% raul = "B" & CStr(j) &
<% raul1 = "N" & CStr(j) &
<tr> </tr>
<% il = il + 1 : j = j + 1
loop%
<% il = il - 1 &
<% k = 0 &
<% do while k < ( 7 - VRDBA(0,0) ) &
<% raul = "C" & CStr(k) &
<% raul1 = "M" & CStr(k) &
<tr> </tr>
<% k = k + 1
loop%
<% end if &
<% il = il + 1
loop%></Table></form>

```

EXTRACT X16 New\_Rau55b1.asp Related to Fig 4.18 (Presentation)

## EXTRACTED FROM NEW\_RAU55B1.ASP

```

<% frmAl = Request.Form("Al") %>
<% VRDBA = Session("rau55b1")
If Not IsEmpty(frmAl) Then
    Set x = Server.CreateObject("Enroll.Enrollment")
    il = 1
    size1 = VRDBA(0,0) + 1
    do while il < size1
        if il = 1 or il = VRDBA(0,0) then
            raul = "A" & CStr(il)
            if il = 1 then
                x0 = x.VB_41_52UPEQ(strInNum, il, 1, VRDBA(1,1), VRDBA(3,11), Request.Form(raul))
            else
                x0 = x.VB_41_52UPEQ(strInNum, 7, 1, VRDBA(1,1), VRDBA(3,11), Request.Form(raul))
            end if
        else
            j = 2
            do while j < VRDBA(0,0)
                raul = "B" & CStr(j)
                raul1 = "N" & CStr(j)
                x0 = x.VB_41_52UPEQ(strInNum, j, 1, VRDBA(1,j), VRDBA(3,j), Request.Form(raul))
                x0 = x.VB_41_52UPEQ(strInNum, j, 2, VRDBA(1,j), VRDBA(2,j), Request.Form(raul1))
                il = il + 1
                j = j + 1
            loop
            il = il - 1
            k = 0
            do while k < (7 - VRDBA(0,0))
                raul = "C" & CStr(k)
                raul1 = "N" & CStr(k)
                x0 = x.Make_new_555(strInNum, Request.Form(raul), Request.Form(raul1))
                k = k + 1
            loop
        end if
        il = il + 1
    loop
    Response.Redirect "Rau_55HTM.asp"
Else%>

```

EXTRACT X17 New\_Rau55b1.asp Related to Fig 4.18 (Interaction)

*Chapter 5***CONCLUDING COMMENTS**

This research paper was motivated by the recognition that, Although legacy systems were designed and built using the best information technology available 30 years ago those systems are becoming too expensive to maintain and the demands for alterations can not be sustained and as a result business opportunities are being lost. [BENN95] The above circumstances added to the proposition that the value of high end PCs workstations and networks are challenging the economics of continuing with IBM mainframe systems justifies the quest for alternatives that are cheaper and more flexibles.

The agenda for this research paper was to attempt to build a model of a mainframe legacy system on a web client/server environment, to itemize the issues emerging when porting the old system to the new environment, and to offer cautionary guidance to future migrators. In chapter four this paper described the development of the model on a web client/server environment, we were able to demonstrate to the user that the new system interface correctly corresponded to the mainframe legacy system in both layout and functionality. First, every interaction (I/O) possible in the legacy system is possible in the new one. Second, the intrascreen navigation possible in the new system was also possible in the mainframe legacy system and viceversa. And finally, the sequence of screens (interscreen navigation) possible in the new interface was also possible in the mainframe legacy system and viceversa. Therefore by building a prototype of a legacy system on a web client/server environment, by presenting a list of the difficulties and issues arising when the prototype was being built and by offering an inventory guide for future migrators This research paper claims to have accomplished its objectives.

There are several avenues for further research in this area. An immediate next step would be to develop an expert system for Converting legacy systems by using AI techniques and ideas that are mature enough to be applied to real-world problems. The tool for converting legacy systems will do an automated reasoning about the legacy system and will generate new constructs in different target languages. For instance from the COBOL-CICS setting to an environment composed of components made in the languages C++, VB, or Java. Since a conversion tool is usually a task-specific the tool can be tailored to convert from the COBOL-CICS setting to the Java environment. And since the expert system will use a process of knowledge acquisition and representation conversion rules. The expert system's knowledge base can be populated with the particular rules of the specific legacy system to be transformed and the particular rules of the specific target language.

This research paper has shown an approach for upgrading existing software to new technology standards. This document has provided a model of a mainframe legacy system on a web client/server environment, and it has granted some ideas that may be useful to future migrators. The next step for this research is to develop an expert system for Converting legacy systems by using AI techniques.

## REFERENCES

- 
- [BASI90] Basili, V., "Viewing Maintenance as Reuse-Oriented Software Development," IEEE Software, Vol. 7, No. 1, January 1990, pp. 19-25
- [BENN95] Keith Bennet, "Legacy Systems: Coping with success", IEEE Software January 1995, pp. 19-23.
- [BERG96] Hal Berghel, "The Web", Communications of the ACM January 1996, Vol. 39 No. 1, pp. 33-40.
- [BIGG89] Biggerstaff T. and C Ritcher, "Introduction to Reusability," In Software Reusability Volume I, Biggerstaff, T. and A.J. Perlis, Eds., Addison-Wesley, New York, NY, 1989, pp. 1-17
- [BIGG90] Biggerstaff T. and C Ritcher, "Reusability Framework, Assessment, and Directions," IEEE Software, Vol. 7, No. 1, January 1990, pp. 19-25
- [BRCO98] L Brown, M. Consens, I. Davis, C. Palmer, F. Tompa "A structured Text ADT For Object-Relational Database", Theory and Practice of Objects Systems, Vol 4(4) 1998, pp. 227-244.
- [BRHE95] O. Bray and M. Hess, "Reengineering a Configuration Management System", IEEE Software January 1995, pp. 55-63.
- [BRWA98] Alan W. Brown, and Kurt C. Wallnau, "The current state of CBSE", IEEE Software September/October 1998, pp. 37-46.
- [CHAR98] Ron Charron, "Training and the Push for Change," JOOP, Vol. 2, No. 6 October 1998, pp. 19-25
-

- 
- [CHEN92] Cheng, J., "Parameterized Specifications for Software Reuse," ACM SIGSoftware Engineering Notes, Vol. 17, No. 4, October 1992, pp 53-59
- [COXB86] Cox, B. and A. Novobiliski, Object-Oriented Programming: An Evolutionary Approach, Addison-Wesley, Reading, MA, 1986
- [DEVR95] G. Dedene and JP De Vreese, "Realities of Off-Shore Reengineering", IEEE Software January 1995, pp. 35-45.
- [FERN98] Luiz Fernandez Capitez "Testing models: the requirements model" JOOP, Vol. 2, No. 3, June 1998, pp. 20-23
- [GRUM88] Gruman, G., "Early Reuse Practice Lives Up to its Promise, "IEEE Software, Vol 5, No. 6, November 1988, pp. 87-91
- [HORO98] Elis Horowitz "Migrating Software to the World Wide Web" IEEE Software May-June 1998, pp. 18-21
- [JOHN88] Johnson, R. and B. Foote, "Designing Reusable Classes," Journal of Object-Oriented Programming, Vol. 1, No. 2, June/July 1988, pp. 23-35
- [KAMC98] Joice M. Kai and James C. Mc Keen, Jr, "Object Oriented Capabilities of Visual Basic," JOOP, Vol. 2, No. 6, October 1998, pp. 46-57
- [KIEL98] Don Kiely, "Are components the future of software", IEEE computer, February 1998, pp. 10-11.
- [KOBO98] Wojtek Kozaczynski, Grady Booch "Component-Based Software Enengineering" IEEE Software September-October 1998, pp. 34-36
-

- 
- [KOTU98] Jeffrey kotula, "Using Patterns to create component Documentation", IEEE Software March/April 1998, pp. 85-92.
- [KRAD98] David Krieger, and Richard M. Adler, "The emergence of distributed component platforms", IEEE computer, March 1998, pp. 43-53.
- [LEFE98] Michael Le Febre, "Designing a persisten object model from partners" JOOP, Vol. 2, No. 1, March-April 1998, pp. 17-28
- [LIBA98] Zheng-Yang Liu, Mike Ballantyne and Lee Seward, "An Assistant for Re-Engineering Legacy Systems", white paper <http://www.spo.eds.com/edsr/papers/asstreeng.html>
- [MAGE97] Ken Magel, "Is it too late to put the user back into HTML", IEEE computer, December 1997, pp. 131-132.
- [MEGA95] E. Merlo, P. Gagne, J. Girard, K. Kontogiannis, L. Hendren, P. Panangaden and R. De Mori. "Reengineering User Interfaces", IEEE Software January 1995, pp. 64-73.
- [MUNS98] John C. Munson "Software Lives too Long", IEEE Software July/August 1998, pp. 18-20.
- [OLLI97] Jon Oler and G. Lindstrom, "Migrating Relational Data to an ODBMS: Strategies and Lessons from a Molecular Biology Experience", ACM SIGPLAN, Vol 32, No. 10 October 1997 pp. 243-251.
- [OLSE98] Michael R. Olsem, "An Incremental Approach to Software Systems Re\_engineering", Journal Softw. Maint: Res. Pract. No 10, 1998, pp. 181-202.
-

- 
- [OUST98] Ohn K. Ousterhout, "Scripting: higher level programming for the 21<sup>st</sup> Century", IEEE computer, March 1998, pp. 23-30.
- [PENG98] C. Peng, S. Chien, J. Chung, A. Roy-Chowdhury and V. Srinivasan. "Accessing existing business data from the world wide web", IBM Systems Journal Vol. 37 No. 1, 1998, pp. 115-145
- [PERS98] Carlo Persio, "Deriving Patterns from Design Principles" JOOP, Vol. 11, No. 6, October 1998, pp. 67-71.
- [PHIL98] Philips J. Gill, "Leading the component Revolution," JOOP, Vol. 2, No. 5, September 1998, pp. 19-25
- [PRDE97] M. W. Price and S. Demurjian, "Analyzing and Measuring Reusability in Object-Oriented Design", ACM SIGPLAN, Vol 32, No. 10 October 1997 pp. 22-33
- [PRIE93] Prieto-Diaz, R., "Status Report: Software Reusability," IEEE Software, Vol. 10. No. 3, May 1993, pp.61-66
- [RUGA98] Spencer Rugaber "Restoring a Legacy Lessons Learned", IEEE Software July/August 1998, pp. 28-33.
- [RUMB98] James Rumbaugh, " The year 2000 crisis and the search for the absolute truth" JOOP, Vol. 2, No. 2, May 1998, pp. 10-127
- [SCHN98] N. Schneidewind, "Preserve or Redesign Legacy Systems", IEEE Software July/August 1998, pp. 14-17.
-

- 
- [SCRO98] Daniel Schwabe, Gustavo Rossi "An object Oriented approach to web-based applications Design", Theory and Practice of Objects Systems, Vol 4(4) 1998, pp. 207-225.
- [SNEE95] Harry M. Sneed, "Planning the Reengineering of Legacy Systems", IEEE Software January 1995, pp. 24-34.
- [SOMM92] Sommerville, I., Software Engineering, 4<sup>th</sup> Edition, Addison-Wesley, Workingham, England, 1992
- [VOAS98] Jeffrey Voas, "Maintaining component-based systems", IEEE Software July/August 1998, pp. 22-27.
- [WEGS97] Eric Wegscheider, "Toward Code-free business application development", IEEE computer, March 1997, pp. 35-43.
- [WOTI95] K Wong, S. Tilley, H. Muller, and M. Storey, "Structural Redocumentation a Case Study", IEEE Software January 1995, pp. 46-54.
-

**APPENDIX A****Tables composition**

INTL\_IVC\_HED

INTL_IVC_NUM: VARCHAR2(6)
CUSTNUM: VARCHAR2(80)
II DATE: DATE
II_SHIP_ADR_LINE1: VARCHAR2(30)
II_SHIP_ADR_LINE2: VARCHAR2(30)
II_SHIP_ADR_LINE3: VARCHAR2(30)
II_SHIP_ADR_LINE4: VARCHAR2(30)
II_SHIP_ADR_LINE5: VARCHAR2(30)
II_SOLD_ADR_LINE1: VARCHAR2(30)
II_SOLD_ADR_LINE2: VARCHAR2(30)
II_SOLD_ADR_LINE3: VARCHAR2(30)
II_SOLD_ADR_LINE4: VARCHAR2(30)
II_SOLD_ADR_LINE5: VARCHAR2(30)
IIHRCOD_TERMS: VARCHAR2(3)
IIHRCOD_COD_TXT: VARCHAR2(30)
IIHCNT_MAX_PG: NUMBER
IIHCNT_P_LINES: NUMBER
IIHCNT_A_LINES: NUMBER
II_CUS_ORD_REF_NUM: VARCHAR2(30)
II_PRE_PAYP_FLG: VARCHAR2(1)
II_COL_FLG: VARCHAR2(1)
II_PP_ADD_FLG: VARCHAR2(24)
II_SHIP_TEXT: VARCHAR2(2)
II_COUNTY: VARCHAR2(2)
II_FRHT_TYPE: VARCHAR2(1)
II_SHIP_NUM: VARCHAR2(3)
II_UNIT_PRC_FLG: VARCHAR2(1)
EMPLNUM: VARCHAR2(4)
INVC_DAT_ORD: VARCHAR2(8)
II_FOT_POINT: VARCHAR2(30)
INVC_ORD_NUM: VARCHAR2(10)
II_SUB_TOTS_FLG: VARCHAR2(1)
II_INVC_PRT_FLG: NUMBER
II_CURR_TYP: VARCHAR2(5)
II_AR_FLG: VARCHAR2(5)
II_AR_CMP_NUM: VARCHAR2(4)
II_AR_DIV_NUM: VARCHAR2(4)
RECNUM: NUMBER

INTL\_IVC\_DET

INTL_IVC_NUM: VARCHAR2(6)
SORDNUM_1_7: VARCHAR2(7)
SORDNUM_LIN_1_3: VARCHAR2(3)
SORDNUM_LIN_4_5: VARCHAR2(2)
SORLQTY_SHI: NUMBER(7)
CUSTORD_DAT: DATE
SORLQTY_SPC_PRC: VARCHAR2(1)
SORLQTY_TRN: NUMBER(2)

INTL\_IVC\_TXT

INTL_IVC_NUM: VARCHAR2(6)
SORDNUM_1_7: VARCHAR2(7)
SORDNUM_LIN_1_3: VARCHAR2(3)
SORDNUM_LIN_4_5: VARCHAR2(2)
II_TXT_SEQ: NUMBER(5)
II_TXFLD_TXT_PRT1: VARCH-AR2(60)
II_TXFLD_TXT_PRT2: VARCH-AR2(4)
II_TXFLD_TXT_DESC: VARCHAR2(31)
II_TXFLD_TXT_PRT2C: VARCHAR2(3)
II_TXFLD_TXT_PRT3: VARCHAR2(14)
II_TXFLD_TXT_USD: VARCHAR2(5)
II_TXFLD_TXT_TOTAL: VARCHAR2(13)
II_TXFLD_TXT_TOT_R: NUMBER(11,2)
SORLQTY_SPC_PRC: VARCHAR2(2)
II_AR_CMP_NUM: VARCHAR2(4)
II_AR_DIV_NUM: VARCHAR2(4)

INTL\_IVC\_SEL\_HD

SORDNUM_1_7: VARCHAR2(7)
II_DATE: DATE
CUSTNUM: VARCHAR2(80)
CUSTNUM_ORD: VARCHAR2(16)

INTL\_IVC\_SEL\_RC

SORDNUM_1_7: VARCHAR2(7)
CUSTNUM: VARCHAR2(80)
SORDNUM_LIN_1_3: VARCHAR2(3)
SORDNUM_LIN_4_5: VARCHAR2(2)
SORLQTY_ORD: NUMBER(7)
SORLQTY_SHI: NUMBER(7)
SORLQTY_II_SHI: NUMBER(7)
ITEMNUM: VARCHAR2(10)
ITEMDES_1: VARCHAR2(31)
ITEMDES_2: VARCHAR2(31)
ITEMDES_3: VARCHAR2(31)
ITEMDES_4: VARCHAR2(31)
SORLPRC: NUMBER(9,2)
SORLPCT_DIS: NUMBER(1,4)
SORLCOD_SPC_PRC: VARCHAR2(1)

INTL\_IVC\_AUTH

EMPLNUM: VARCHAR2(4)
IIAINTL_INV_TITLE: VARCHAR2(65)
EMP_PASSWORD: VARCHAR2(18)
EMP_EMAIL: VARCHAR2(18)

CM\_MSTR\_EXT

U#CM_ADDR: VARCHAR2(80)
LAST_NAME: VARCHAR2(50)
FIRST_NAME: VARCHAR2(50)
MIDDLE_INITIAL: VARCHAR2(10)
COMPANY_NAME: VARCHAR2(50)
E_MAIL_ADDRESS: VARCHAR2(30)
NAME_PREFIX: VARCHAR2(4)
NAME_SUFFIX: VARCHAR2(4)

cm\_mstr
