2004

Benchmarking VisualStudio.NET for the development and implementation of a manufacturing execution system

Amit S. Rege

Rochester Institute of Technology

Benchmarking VisualStudio.NET for the Development and Implementation of a Manufacturing Execution System

A Thesis

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Industrial Engineering

in the

Department of Industrial & Systems Engineering

Kate Gleason College of Engineering

by

Amit S.Rege

M.S, Industrial & Systems Engineering, Rochester Institute of Technology, 2004

October, 2004

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The **M.S. Degree Thesis of Amit S.Rege** has been examined and approved by the thesis committee as satisfactory for the thesis requirement for the **Master of Science degree in Industrial & Systems Engineering**

Approved by:

Sudhakar Paidy

Dr. Sudhakar R. Paidy, Thesis Advisor
Professor, Industrial & Systems Engineering Department
Kate Gleason College of Engineering

James B. Taylor

Dr. James B. Taylor, Committee Member
Associate Professor, Industrial and Systems Engineering Department
Kate Gleason College of Engineering

Permission Granted

Benchmarking VisualStudio.NET for the development and implementation of a Manufacturing Execution System

I, Amit S.Rege, hereby grant the permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for any commercial use or profit.

Signature of Author:

Amit S. Rege

Amit S.Rege
Industrial & Systems Engineering Department
Kate Gleason College of Engineering

Dedication

This thesis is dedicated to my parents

Mr. Sharad S. Rege

And

Mrs. Beena S. Rege

who have given me the opportunity to be who I am today.

To my little niece Ria Rege (born 12/31/2003), Thank you for coming into the world!

Acknowledgement

This dissertation could not have been possible without Dr. Sudhakar Paidy who not only served as my advisor but also encouraged and challenged me throughout my academic program. He patiently guided me through the research process, never accepting less than my best efforts, and above all, he always believed in me.

I also would like to thank Dr. James Taylor for supporting my work and providing me with valuable suggestions for improving my work.

To fellow graduate research assistants, Pradip Chandratre and Bharani Govindasamy, I thank them for their support throughout the course of this study.

Also, Prakash Gandhi and Adwait Palsule, as their respective theses were invaluable sources of information.

To my roommates, Amit Chaudhary, Bhavin Vora, Siddharth Mestry, I thank them for believing in me and providing constructive criticism to prepare me for my thesis.

I also would like to thank my department head, Dr. Jacqueline Mozrall, our staff assistant, Marilyn Houck, and the all the members of the department, for providing a very motivating work environment.

Last but not the least, I thank my brothers Rajesh and Anil, and my sisters-in-law Dipali and Krithika, who gave me invaluable support in times of need and always believed in me.

Abstract

The focus of this thesis is to show the utility of Microsoft's' .NET framework in developing and implementing a MES system. The manufacturing environment today, more than ever, is working towards achieving better yields, productivity, quality, and customer satisfaction. Companies such as DELL are rapidly outgrowing their competition due to better management of their product lifecycles. The time between receiving a new order to the time the final product is shipped is getting shorter. Historically, business management applications such as Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems have been implemented without too much importance given to the operational and shop floor needs. The fact is that these business systems can be successful only when they are properly integrated with real-time data from the shop floor, which is the core of any manufacturing set-up. A Manufacturing Execution System or a MES is this link between the shop floor and the top floor. MESA international defines MES as "Systems that deliver information enabling the optimization of production activities from order launch to finished goods" Thus, a MES provides the right information to the right people at the right time in a right format, to help them make well-informed decisions. Thus, a necessity for an efficient MES is high capability of integration with the existing systems on the operational level. This is where Microsoft's' VS.NET fits in.

Microsoft defines .NET as "***A set of software technologies for connecting information, people, systems and devices***". The vision of .NET is to enable the end user to connect to information from any place at anytime, using any device and in a manner that is independent of the platform on which the service is based. The building block of the .NET framework is the Common Language Runtime or CLR, which is capable of converting data from its original format into a format understandable to .NET and then use that format to interface with its client. This feature that .NET provides holds the key in the context of a MES development and implementation.

The aim of this applied research is to design a MES using VS.NET to control the working of a Flexible Manufacturing System (FMS) namely CAMCELL. The architecture used for the MES will then be gauged against an MES implementation done previously using a Siemens' PC-based automation technology and Visual FoxPro. This study will integrate the Siemens' technology with the .NET framework to enhance the resulting MES efficiency. The shop floor details or the real-time data collection will be done using the databases from WinCC and data aggregation and manipulation will be done within the .NET framework. The software architecture used for this study will achieve vertical integration between the CAMCELL ERP layer, the MES layer and the Control layer. The study will demonstrate how the data stored in a high level ERP database can be converted into useful information for the control layer for process control and also how real-time information gathered from the control layer can be filtered into useful information up to the ERP layer to facilitate the decision making process. VS.NET user interface screens will be proposed to support these activities. The performance of the proposed architecture will be compared to that from previous studies, thus benchmarking VS.NET for the implementation of the MES.

Table of Contents

No.	Div.	Section	Page No.
1.		Introduction.	1
2.		Introduction to Microsoft .NET Framework	
	2.1	Integrated Development Environment	4
	2.2	.NET evolution	5
	2.3	XML & Web Services	6
	2.4	.NET Platform	8
	2.5	What is Microsoft.NET	10
	2.6	The Common Language Runtime	11
	2.7	The .NET Concept	12
	2.8	Visual Studio.NET environment	13
	2.9	Windows Forms	17
	2.10	Windows Forms Control Hierarchy	18
	2.11	GDI+	20
	2.12	Using Object Databases in .NET	21
3.		Manufacturing Execution Systems	
	3.1	Evolution of Manufacturing Systems	25
	3.2	Enterprise Resource Planning	27
	3.3	The Control Layer	28
	3.4	A channel for making a better decision	29
	3.5	Manufacturing Execution Systems	31
4.		Work done in the CAMCELL	
	4.1	The CAMCELL	42
	4.2	Systems integration using Siemens PC based automation technology	45
	4.3	A manufacturing execution system using Siemens PC based automation technology	45
5.		Data Access Standards	
	5.1	Introduction to Data Access	47
	5.2	Understanding Open DataBase Connectivity	49
	5.3	The OLE DB revolution	52
	5.4	The advent of ADO	55
	5.5	Migration to ADO.NET	58
	5.6	Glossary	62
6.		The MESASI database	
	6.1	The real-time MES database	66
7.		Manufacturing Execution System at the Advanced Systems Integration Lab	72
	7.1	Application Overview	72
	7.2	The Client / Customer Module	75
	7.3	The Management Module	78
	7.4	The Operator Module	82
	7.5	The System Administrator Module	85
8.		Benchmarking VS.NET as an application development software	
	8.1	C sharp (C#)	89
	8.2	Interoperability of .NET	91
	8.3	Application Deployment	93

	8.4	Connection with a backend server	95
	8.5	Transactions	99
	8.6	GDI+	99
	8.7	Query builder to generate SQL queries	100
	8.8	XML integrity with .NET framework	101
	8.9	The simplicity for application development	101
	8.10	Remoting in .NET	102
	8.11	The DataSet object in the .NET framework	102
9		Conclusion	104
10		APPENDIX A: The Picture Slide Application Tutorial	106
11		APPENDIX B: Database Normalization	135

List of Figures

No.	Div.	Section	Page No.
2.		Microsoft .NET Framework	
	2.1	A sample IDE for visual C++	4
	2.2	Windows DNA Architecture	5
	2.3	The .NET concept	10
	2.4	Utility of CLR	12
	2.5	Structure of solutions and projects	14
	2.6	Windows forms controls hierarchy	18
	2.7	Sample screen for a Windows Form	20
3.		Manufacturing Execution Systems	
	3.1	Manufacturing Resource Planning	26
	3.2	Concept of an ERP system	27
	3.3	The Control Layer	28
	3.4	Hierarchy of information systems in a manufacturing industry	30
	3.5	Time benefit of a MES implementation	31
	3.6	MES, an enterprise flow model	32
	3.7	MES context model	34
	3.8	MES functional model	35
	3.9	The corporate benefits of MES	39
	3.10	The operational benefits of MES	39
4.		Work done in the CAMCELL	
	4.1	Schematic of the CAMCELL material	44
	4.2	The CAMCELL hardware architecture	44
	4.3	Data Flow for the CAMCELL	45
5.		Data Access Standards	
	5.1	Basic concepts of a DBMS	47
	5.2	Need for a data access standard	48
	5.3	ODBC architecture	49
	5.4	OLE DB object model	53
	5.5	OLE DB component interaction	54
	5.6	ADO object model	56
	5.7	Where AOD.NET fits in	58
	5.8	ADO.NET architecture	59
6.		The MESASI database	
	6.1	Three tier architecture	66
	6.2	The SEQ file	69
7.		Manufacturing Execution System at the Advanced Systems Integration Lab	72
	7.1	Data acquisition with Siemens WinCC	72
	7.2	Client-Server computing model	73
	7.3	MES application login screen	74
	7.4	Customer main menu	75
	7.5	View customer order	76
	7.6	Place an order	77
	7.7	Manager main menu	78
	7.8	Summary of orders	79
	7.9	OEE statistics	80

	7.10	Operator main menu	82
	7.11	Production schedule	83
	7.12	Administrator main menu	85
	7.13	Customer information	86
	7.14	Part information	87
8.		Benchmarking VS.NET as an application development software	
	8.1	Comparative code architecture of VS.NET vs. JAVA	91
	8.2	CLS and CLR	91
	8.3	Interoperability with .NET	92
	8.4	Deployment in legacy windows systems	93
	8.5	Deployment in .NET framework	93
	8.6	GAC in .NET framework	94
	8.7	Creating a DSN	95
	8.8	Connection designer in Visual FoxPro	95
	8.9	SqlConnection object in the data tab of the toolbox in VS.NET IDE`	97
	8.10	Identify the service provider and establish the connection directly to the SQL server	98
	8.11	Query builder in VS.NET	100
	8.12	Various applications that can be developed in .NET	101
	8.13	Remoting in .NET	102
	8.14	The DataSet object in .NET	102
10.		APPENDIX A: The Picture Slide Application Tutorial	
	a.1	The picture slide application	106
	a.2	Opening a new project in VS.NET IDE	107
	a.3	Define the name for the solution and the project	108
	a.4	An empty windows form	108
	a.5	Properties window to change the properties of the form	109
	a.6	The view pull down menu for the properties window	110
	a.7	mainMenu from the Toolbar	112
	a.8	The 'File' menu item for the mainMenu	112
	a.9	The 'Size' menu item for the mainMenu	113
	a.10	Images for ImageList	114
	a.11	Image Collection editor for the ImageList	115
	a.12	Add the appropriate images for the ImageList	116
	a.13	List of selected images with the respective index values	116
	a.14	Toolbar for the application	117
	a.15	The 'open' Toolbar button	118
	a.16	The desired buttons, images and the tips using the editor	118
	a.17	The ButtonSize property for the toolBar	119
	a.18	Open File Dialog Box	120
	a.19	The 'Timer' from the Components tab	121
	a.20	The Status Bar	122
	a.21	The StatusBarPanel Collection Editor	123
	a.22	A second form for the 'About Box'	124
	a.23	The RichTextBox for the About Form	125
	a.24	Event handling for the toolbar	126
	a.25	Code editor for the toolbar 'onclick' event	127
	a.26	Event handling from the properties window	128
	a.27	Events for the Picture Form	130
	a.28	Variable declaration using the class view	132
	a.29	Using the field wizard from the Class view	132

List of Tables

No.	Div.	Section	Page No.
10.		APPENDIX A: The Picture Slide Application Tutorial	
	a.1	The properties for the PictureForm	110
	a.2	The properties for the mainMenu	111
	a.3	Properties for the ToolBar buttons (ToolBarButton Collection Editor)	119
	a.4	Properties for the Statusbar sections	123
	a.5	The code for the sub menus	129
	a.6	Location of the event property code for PictureForm	130
11		APPENDIX B: Database Normalization	
	b.1	Table that needs to be in 1NF	135
	b.2	Order Table in 1NF	136
	b.3	Table that needs a 2NF conversion	136
	b.4	customerorderheader table	137
	b.5	customerorderdetail table	137
	b.6	Order table that needs a 3NF conversion	138
	b.7	Order Table	138
	b.8	Customer table	138

List of Tables

No.	Div.	Section	Page No.
10.		APPENDIX A: The Picture Slide Application Tutorial	
	a.1	The properties for the PictureForm	110
	a.2	The properties for the mainMenu	111
	a.3	Properties for the ToolBar buttons (ToolBarButton Collection Editor)	119
	a.4	Properties for the Statusbar sections	123
	a.5	The code for the sub menus	129
	a.6	Location of the event property code for PictureForm	130
11		APPENDIX B: Database Normalization	
	b.1	Table that needs to be in 1NF	135
	b.2	Order Table in 1NF	136
	b.3	Table that needs a 2NF conversion	136
	b.4	customerorderheader table	137
	b.5	customerorderdetail table	137
	b.6	Order table that needs a 3NF conversion	138
	b.7	Order Table	138
	b.8	Customer table	138

Chapter 1: Introduction

Modern manufacturing is done using automated processes, machines and computer controls that are manually supervised and have evolved into sophisticated systems that efficiently integrate a large number of resources. Today, the speed and cost with which new products are developed and delivered to the market is often the chief determinant of competitive success. Development of a system can be defined as an integration of machines, robots and human resources that perform one or more operations to transform raw material into a final product. The success of a manufacturing system depends on the timely access to product information from all stages of the product manufacturing cycle. Thus, the choice of a development environment for effective support systems for manufacturing becomes a key aspect in its success. The development environment needs to be very efficient and should have the ability and flexibility to adapt to the ever-changing needs and features of the manufacturing support systems today. Thus, there is a great need for two-way information flow, between the management support systems and manufacturing. This information exchange is achieved through an effective Manufacturing Execution System, or MES.

This applied research work aims at understanding the importance of an effective MES in the context of the CAMCELL that has been used in the ASI Lab at Rochester Institute of Technology. The goal of this study was to benchmark Microsoft's VS.NET software development platform with the Visual FoxPro software platform as a candidate for developing a distributed information system application such as the MES.

The precursors to this work are two research theses [1, 2] used to examine the utility of the Siemens PC based automation technology within a manufacturing environment. One focused on data acquisition from the CAMCELL using the Siemens Step 7 and soft PLCs and the other used WinCC to place the data into a real time database as well as to pass the information to a MES developed in Visual FoxPro.

The application developed for this work was named MESASI, an acronym for '**M**anufacturing **E**xecution **S**ystems of the **A**dvanced **S**ystem **I**ntegration lab'. The front end application is developed using the Visual Studio.NET (VS.NET) platform while Microsoft's SQL server hosted the back end data base. The SQL server holds the real time data from the CAMCELL and that is massaged into useful information for the business management layer using the VS.NET user interface.

Chapter 2 presents an overview of the Visual Studio.NET platform and its various elements. It points out how VS.NET has the capability of integration with various languages. It gives an overview of the Common Language Runtime or CLR that is used by the .NET Framework. It also introduces the concept of 'projects and solutions' in .NET. An application for an 'Image Viewer' was developed to present some key features of VS.NET development environment (Appendix A).

Chapter 3 describes the concept of a MES and presents the various benefits of the MES. It describes the various elements that are needed for an MES and those that depend on the MES. It explains the functional model and the context model of a MES. It presents the key elements of the control layer and the ERP layer and explains how the transfer of the information between these two layers is essential for a successful MES application. Chapter 4 provides a brief description of the CAMCELL and the two related works cited earlier.

Chapter 5 explains the progression of the data access standards from the ODBC standard to the present ADO.NET standard. It also presents the key benefits of the ADO.NET standard. Chapter 6 describes the MESASI database and gives an overview of the data tables that were created for the database and how the information was organized. Appendix B presents the normalization principles for organizing data into a database.

During the development of the MESASI application, four separate modules were generated for illustrating the functionality of the MES. These four modules are the system administrator module, the customer module, the management module and the operator module. For each application module, two forms were designed that would show the adaptability of the VS.NET software and present the information needed to be viewed by the end-users of that particular module. Chapter 7 describes the MESASI application with a step by step explanation of each of the customized screens developed for each of the four modules.

Chapter 8 contrasts the features of Visual Studio .NET with those of Visual FoxPro. The interoperability of .NET and the ease of deployment are explained. Chapter 9 presents the conclusions from the development of the MESASI application and recognizes the Visual Studio.NET software as a very good choice for developing a Manufacturing Execution System.

Reference:

1. Adwait Palsule, *"Systems Integration using Siemens PC based automation technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2002.
2. Prakash Gandhi, *"A Manufacturing Execution System using Siemens' PC Based Automation Technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2003.

Chapter 2: Introduction to Microsoft .NET Framework

2.1 Integrated Development Environment (IDE)

An "Integrated Development Environment" is typically a software package that consists of a code text editor (with a defined syntax), a compiler, a builder and a debugger. Typically, each programming language has an independent IDE, for example visual C++ (see Figure 2.1). Many times, the IDE is accompanied by a built-in GUI, which essentially makes the job of the end user much simpler. An IDE is also referred to as an "Integrated Design Environment" or "Integrated Debugging Environment" [1].

An IDE is both interactive as well as an integrated system. It does allow the user to interactively change the content of a program and thus have control over the output. It is integrated in a way that the text editor and the development platform are closely coupled.

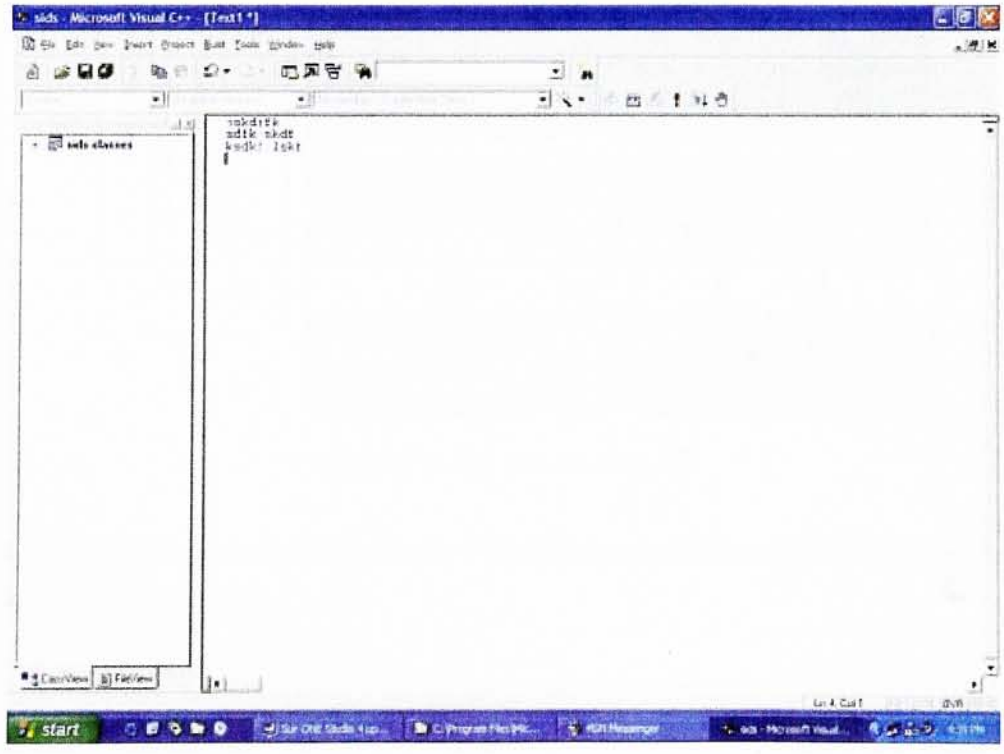

Figure 2.1 A sample IDE for visual C++

2.2 . NET evolution

In 1999 Microsoft introduced its **Windows Distributed interNet Application** (DNA) architecture with the imminent arrival of Windows 2000 [23]. This was essentially an architecture that described how to build three tier applications. (see Figure 2.2).

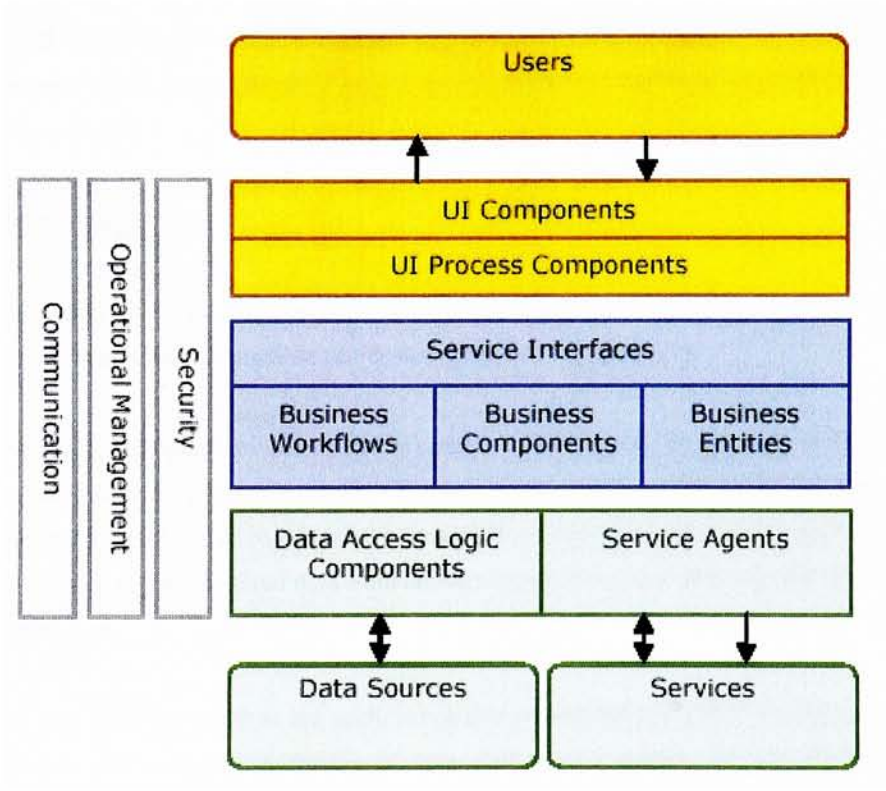

Figure 2.2 Windows DNA Architecture

The .NET platform is built extensively on the Windows DNS architecture by introduction of many new languages and technologies.

2.3 XML & Web Services

2.3.1 What is XML?

XML stands for "Extensible Markup Language". It is a markup language that contains and provides structures information. The XML specifications define a standard way of representing and indicating these structures that are contained in a document. XML is considered by most to be the standard or universal language or platform for data exchange over the Internet [2].

2.3.2 What is a WEB Service?

The WWW is widely used for communication between different applications and smart machines or computers. The interfaces and software that make this possible can be described as "Web Services'.

Web Services provide a general platform that is used to communicate between applications and transfer information between them. Microsoft describes Web Services as "*Discrete units of code capable of performing a limited set of tasks*" [3]. They are based on XML and can respond to any platform, no matter what the programming language is at the backend. They can also use the required data from another application, regardless of what specs go into the working of that application.

In general, it can be said that Web Services are applications that provide great utility to its end user by simplifying their tasks as that user is able to use the capabilities of other applications without having access to its respective mainframe and most importantly independent of the programming platform used by the other application.

As an example, assume two companies want to exchange information. One company deals with the SQL database and another has the information in an ORACLE Database. Using web services would make their life much simpler, as it will act as the mediator that converts or interprets the data to its end user depending on the environment that the end user has and is independent of the environment at the provider station.

2.3.3 SOAP

Simple Object Access Protocol: XML Services implemented by SOAP allow applications to share data as well as invoke methods and properties of other remote applications without any prior knowledge of the remote applications architecture [23].

2.3.4 VS.NET Drives XML Web Services

In this phase of computer evolution, XML Web services, promises to have the greatest impact and most rapid adoption of any previous technology.

Microsoft's support and integration of XML Web services is encapsulated in the .NET platform and strategy. . NET consists of clients, servers, and services—all held together with a common hub: Visual Studio .NET (VS.NET). VS.NET and the wealth of supporting tools and components will drive this evolutionary phase towards XML Web services [8].

2.4 .NET Platform

At the highest level, the .NET platform includes the following [23]:

- . NET Framework.
- . NET building Block Services.
- . NET Enterprise Servers
- Visual Studio.NET

2.4.1 . NET Framework

The .NET framework is based on CLR libraries that are available to any of the .NET languages, across all the tiers of the application.

The .NET framework is made up of the following:

- **Common language runtime** – Run time environment for all languages managing memory, garbage collection and security.
- **. NET Framework Class Library** – Extensive and comprehensive new class library based on namespaces acting as containers for class hierarchy.
- **ADO.NET (data and XML)** – Data Access. A rewrite of ADO claiming to do pretty much the same as ADO, albeit a bit better.
- **ASP.NET (Web Forms and Services)** – Compiled code based on web Forms and event driven programming for web pages.
- **User interface components** – Uses win Forms for windows applications.

2.4.2 . NET Building Block Services

. NET building block services are a programmable user centric set of XML distributed services that can be used from any platform that supports SOAP. Building Block services include Passport for user identification and other services for message delivery, file storage, user preference management, notification, directory, and search or software delivery. They are dependant on SOAP and XML.

2.4.3 . NET Enterprise Servers

The .NET Enterprise Servers consist of a range of .NET branded infrastructure/server applications optimized for building, deploying and managing data and XML web services within the .NET platform.
These include business process, data, scale-out and integration services and solutions such as the following:

- **Application Center 2000** – Scale out solutions.
- **BizTalk Server 2000** – XML based business process orchestration.
- **Host Integration Server 2000** - Mainframe data and application access.
- **Mobile Information Server 2001** – Enables use of applications by mobile devices.
- **SQL Server 2000** – RDMS to store and retrieve structured XML data.

2.4.4 Visual Studio .NET

The Visual Studio .NET development environment consists of the following:

- Visual Basic .NET
- Visual C# .NET
- Visual C++ .NET
- Jscript .NET

2.5 What is Microsoft.NET??

Microsoft defines .NET as "***A set of software technologies for connecting information, people, systems and devices***" [4]. It enables high-level software integration through use of web services (small & discrete building block applications) that can connect to each other as well as other large applications over the Internet.

The .NET technology enables the creation and use of XML-based applications, processes, and Web sites as services that share and combine information and functionality with each other by design, on any platform or smart device, to provide tailored solutions for organizations and individual people. It is built on open standards and thus encompasses all programming languages. The .NET vision is essentially to enable its end user to connect to information to anywhere, anytime, using any device and is independent of the platform that the useful service is based on.

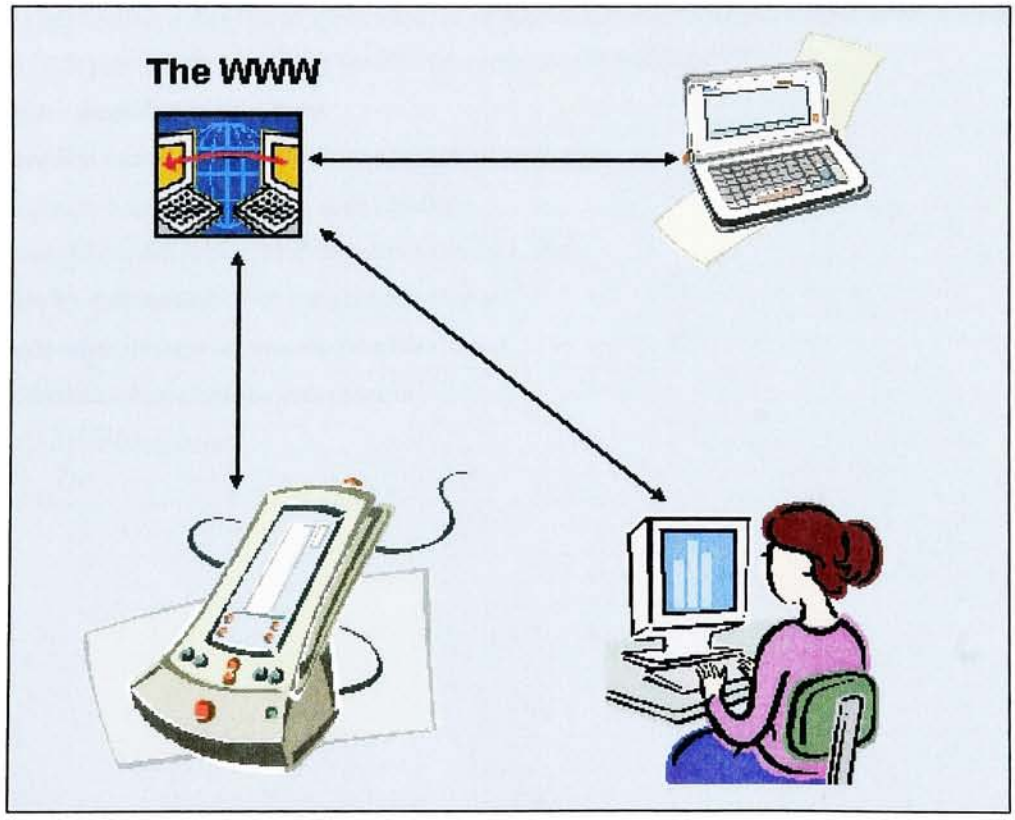

Figure 2.3 The .NET Concept [4]

Thus, using the .NET technology, an individual or a business can exchange data over the Internet and use applications over the Internet without actually having the applications installed on their machine.

2.6 The Common Language Runtime:

The 'Common language runtime' is also referred to as "CLR". It is the environment in which all applications or programs run in VS. Net [5]. This is the common runtime that is used by all languages. This implies that the individual components can be written in and used by any other language as all the languages represent the same types and objects in a similar manner. Also, the languages can use the same API to access the platform services.

This amounts to a set of common services that can be accessed from a variety of object languages. These services will now be executed by intermediate code that is independent of the underlying architecture, providing transparent and seamless interoperability between a variety of applications and services.

The Common Language Runtime (CLR) provides a solid foundation for developers to build various types of applications. Whether you are writing an ASP.NET application, a Windows Forms application, a Web Service, a mobile code application, a distributed application, or an application that combines several of these application models, the CLR provides the following benefits for application developers [2.11]:

- Vastly simplified development
- Seamless integration of code written in various languages
- Evidence-based security with code identity
- Assembly-based deployment that eliminates DLL Hell
- Side-by-side versioning of reusable components
- Code reuse through implementation inheritance
- Automatic object lifetime management
- Self-describing objects

2.7 The .NET Concept:

The vision of Microsoft while producing .NET was to integrate applications over the net or www. This technology is capable of taking the best of all worlds and utilizing it for running their own applications. Certain languages may have certain good qualities and some drawbacks. Thus, for every application that is developed, there will always be a certain programming language that will best suit that application. In today's industry though, integration of more than one dissimilar application is a must to be successful. Thus, there is a scenario in which a business has to take different applications and make these applications talk to each other or interact and most importantly exchange useful information. Doing this in a conventional manner would be very costly and time consuming. It would need different softwares that support those applications, as they require a separate IDE for each different language. Not to mention the brain that goes behind the programming. This does add up in the set-up cost for the whole application (one which is the integration of these small applications) at a macro level.

Fortunately, .NET provides a solution to these questions. It provides a facility of not only using the different applications remotely without actually having these installed on the local machine, but it also provides a single development environment for all the programming languages and these then are interpreted by the CLR. Thus, these different applications get integrated to be useful for the final user. (see Figure 2.4)

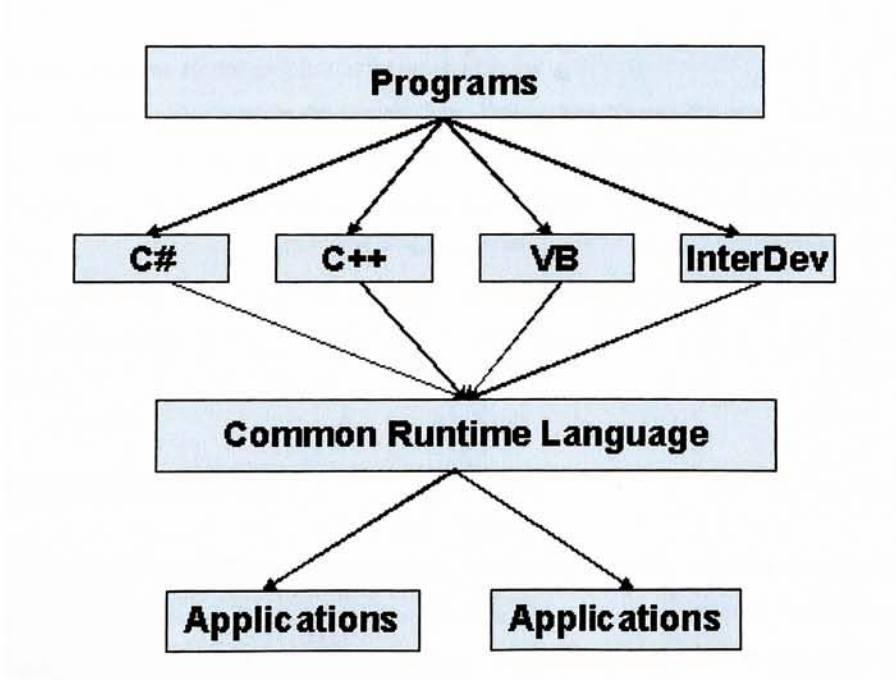

Figure 2.4 Utility of the CLR

2.8 Visual Studio.NET environment

Visual Studio.NET is an IDE platform that is a culmination of over a decade of work on Visual basic, Visual C++, Visual Interdev, and Visual J++. This provides the user a common IDE for all these different languages and has the capability of working effectively with all the different applications either separately or together. Visual Studio.NET is commonly referred to as "VS. Net".

VS.NET consists of two basic units [6]:

- **The Solutions.**
- **The Projects.**

In order to develop any application in VS.NET, a solution is required, and the solution consists of one or more projects. Thus, conceptually a solution is like a briefcase for all the applications that are created in VS.NET. Everything that is done in VS.NET revolves around these two concepts, and thus understanding these concepts becomes all the more important in order to extract the correct utility out of this tool.

2.8.1 Solution:

A solution essentially contains all the projects associated with the application or the projects that are included in the application. These projects in turn contain the project files. Due to this reason, the case structures in the applications become very important to configure. At any given instance of VS.NET, there can only be one solution that is open and operational, and there can be multiple projects that are open i.e. are somehow integrated into that opened solution. One solution cannot import another solution or include another solution into its structure. This condition, however, it does not hold true for the projects. The same project can be a part of more than one solution. The solution not only contains all the projects for the application but also the information of the dependencies between them. (see Figure 2.5).

2.8.2.1 Creating a solution in VS.NET

While creating a solution in VS.NET, it is always advisable that the directory structure and the solution structure are the same. Thus, for each individual solution there should be a directory, and then there should be separate sub-directories for each project that is included in the solution. The VS.NET IDE provides this feature by giving an option to the user of selecting a 'create directory for solution' checkbox, while the user is prompted to give a name to the solution that he is about to create, by default tough, it stores the solution in the same directory as the project). At this point, the user is also prompted to choose the platform that he/she wishes to use (Visual J++, or Visual C++).

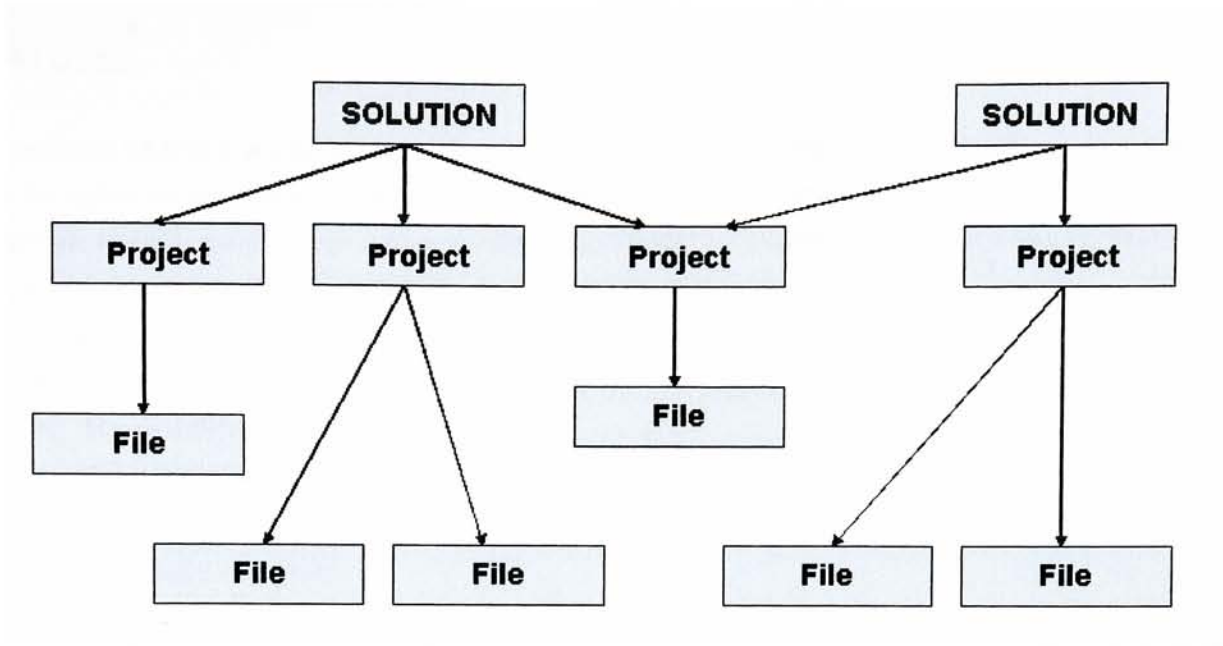

Figure 2.5 Structure for Solutions and Projects

2.8.2 Project:

A project in VS.NET is similar to a project in any other development environment. It contains the source code files for the application and then compiles the source code into an executable (*.exe) or a dynamic link library (*.dll). There are several types of projects that are included in the VS.NET environment. VS.NET2003 does list 90 distinct types of project files but they can be broadly classified into six types or groups.
These six groups are as follows:

- Managed local projects.
- Managed web projects.
- Smart device projects.
- Unmanaged local projects.
- Unmanaged web projects.
- Setup projects.

2.8.2.1 Managed Local Projects:

The managed local projects create a .NET assembly. It can be written in C#, J#, VB.NET or MC++.

2.8.2.2 Managed Web Based Project:

For the managed web based projects, the output is used over a web server, which can either be a local machine or a remote server. This application would need more than the .NET assembly to run, other files contained might include image files, .css files and /or .html files that also must be present on the server. In effect, the entire project is present on the web server.

2.8.2.3 Smart Device Project:

Smart device project, as the name suggests, is a very 'smart' application where the project allows you to build applications that can be run on palmtop devices. These target only C# and VB.NET. These either create a windows application, a class library, a non graphical application or an empty project. VS.NET ships with an emulator that enables the user to test and run the application on a computer without actually having a PDA.

2.8.2.4 Unmanaged Local Project:

Unmanaged local projects build unmanaged executable files. These fall into three groups depending on the library that they use:

- Active Template Library (ATL) projects.
- Microsoft Foundation Class (MFC) projects.
- Win32 projects.

2.8.2.5 Unmanaged Web based Project:

Two ATL web based projects, ATL server project and ATL web service, let the user build web applications and web services respectively. Both projects build ISAPI extensions using ATL server classes. These projects are managed over a web server.

2.8.2.6 Setup and Deployment:

These projects allow the user to create Microsoft Installer Files (.msi) for the use of any project.

2.9 Windows Forms

Microsoft Windows Forms is the set of classes in the .NET Framework that enables the rapid development of powerful smart client applications [13]. The new programming interface for writing windows applications with GUI's is called Windows Forms. Windows Forms are the basic API (Application Programming Interface) used by the .NET framework. In the past many web developers have been using ActiveX controls to provide a very sophisticated environment for the end user. Using the .NET framework, developers can now make much lighter and compact, but also secure objects for the client side and most importantly, it is integrated within the Internet Explorer.

Windows Forms is a name that has been given to that part of the .NET framework class library that is used to build rich client side applications. Windows Forms controls are all based on the class **System.Windows.Forms.Control** [5]. The core element of Windows Forms is the control class, and it is also the foundation of all the user interface applications. Windows Forms can be written in any language that is supported by CLR.
Essentially, it is a new Forms package that enables developers building Windows-based applications to take full advantage of the rich user interface features available in the Microsoft Windows operating system. Windows Forms is part of the new Microsoft .NET platform and leverages many new technologies, including a common application framework, managed execution environment, integrated security, and object-oriented design principles. In addition, Windows Forms offers full support for quickly and easily connecting to Web Services and building rich, data-aware applications based on the ADO+ data model. With the new shared development environment in Visual Studio .NET, developers will be able to create Windows Forms applications using any of the languages supporting the .NET platform, including Microsoft Visual Basic .NET and C#.

2.10 Windows Forms Control Hierarchy

Most of the controls in the **System.Windows.Forms** namespace are actually derived from **Control**. **ScrollableControl** adds support for scrolling the client area of a window. Generally, that scrolling support is accessed through **ContainerControl**, which derives from ScrollableControl and adds support for managing child controls, focus issues, and tabbing. Derived from ContainerControl is **Form**, Windows Forms' top-level control, which has properties to control caption bars, system menus, non-rectangular windowing, and default controls. Also derived from ContainterControl is **UserControl**, which is the base class for controls that developers can build. UserControl is intended to host other child controls but to be exposed as a single unit to outside clients. UserControl and Form both have visual designers in Microsoft Visual Studio .NET and you will find project items for adding and designing classes derived from them [14].

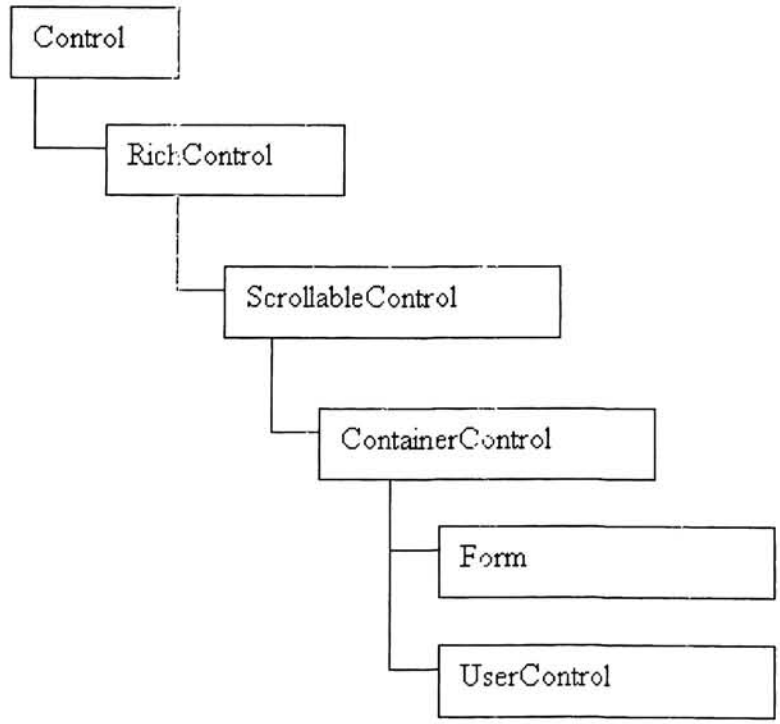

Figure 2.6 Windows Forms Control Hierarchy

A few of the benefits of Windows Forms are:

- Extending the Microsoft Visual Basic rapid application development experience to all of the programming languages supported by the .NET Framework (more than 20).
- Providing developers with rich, powerful, extensible libraries for user-interface controls and graphics.
- Providing dramatically improved support for localization, printing, layout, and usability.
- Making it easy for developers to tie XML Web services into their smart client applications.
- Dramatically reducing total cost of ownership through 'no-touch' deployment, which eliminates 'DLL hell' and makes it possible for smart client applications to be deployed and updated using a remote Web server.

2.11 GDI +

Microsoft Windows GDI+ is a class-based application-programming interface (API) for C/C++ programmers. It enables applications to use graphics and formatted text on both the video display and the printer [14, 15]. Applications based on the Microsoft Win32API do not access graphics hardware directly. Instead, GDI+ interacts with device drivers on behalf of applications. GDI+ is also supported by Microsoft Win64. GDI+ can be used in all Windows-based applications. GDI+ is new technology that is included in Windows XP and the Windows Server 2003. Windows Forms takes full advantage of GDI+, Microsoft's next generation 2-D graphics system. The graphics-programming model in Windows Forms is fully object-oriented and the assorted Pens, Brushes, Images, and other graphics objects are designed following the same ease-of-use guidelines as the rest of the .NET Framework. Developers can now include great new drawing features, such as alpha blending, color gradients, textures, anti-aliasing, and image formats other than bitmaps. When coupled with the Windows 2000 operating system's layered and transparent windows features, developers can create richer, more graphical Win32 applications with much less effort.

When a control's **OnPaint** event is fired, the **System.Drawing.Graphics** object that is accessible from the **PaintEventArgs** is a GDI+ graphics object. All of the operations that the graphics object can perform execute through GDI+.

Figure 2.7 Sample Screen for a Windows Form

2.12 Using Object Databases in .NET

The Microsoft .NET platform is quickly finding market share in consulting companies and large IT shops that build object-oriented enterprise applications. Its object-oriented languages, Web page event-driven programming model, and ease of development all make .NET a powerful platform on which to build object-oriented systems [7].
Most of these systems, however, have their carefully designed object-oriented architecture stopped cold when it hits the database and data-access programming objects. Traditional relational database management systems (RDBMS) simply do not end themselves very well to object-oriented programming. Thirty percent of an application's code alone is used for mapping an application's object-oriented design to the database's relational model.
There is another way, however, and it goes back to a technology that started in the 1990s. Object databases (ODBMS) are a solution to the tedium of relational database mapping code. Though not for every project, object databases are a powerful element in an enterprise architect's bag of tricks.

2.12.1 Relational Databases

The relational database is, of course, well known by most architects and developers working with typical enterprise, line-of-business, or e-commerce applications. A workhorse of the industry since the mid-seventies, the RDBMS is well understood and is easily adapted to a variety of applications. Developed around the idea that information can be stored in flat two-dimensional tables and then related to other pieces of data in other tables through keys, the relational database is a proven technology.

2.12.2 Object Databases

The object database (ODBMS) is that different way. Object databases moved beyond the research labs and into commercial applications in the mid 1990s and have steadily grown in acceptance. Offering a fundamentally different way of designing the persistence layer for an application, object databases are an often-overlooked element in object-oriented systems.

The core idea behind object databases is that you use your data in the same fashion as you store it. Developers like this! There is no mental paradigm shift between the time the data is used for a business calculation and the time it is persisted. The developers simply use their objects as indicated in the class diagrams, and the objects know how to persist themselves at the end of an operation.

Object databases typically do not have the large overhead in terms of code maintenance and administration that a relational database carries. At the same time, typical relational database features such as transactions, concurrency, atomicity, online backups, and other important administrative tasks are supported by most object databases. Major vendors in the field include Matisse Software Inc., Objectivity Inc., Poet's FastObjects, Computer Associates, and eXcelon Corporation. There are a few open-source implantations as well. The industry is constantly changing, and at the moment it is mostly Java focused. Matisse is the only major vendor to presently offer a .NET binding to its database, though FastObjects (a spinoff from Poet) is readying its for release shortly.

Reference:

1. From Wikipedia, The free encyclopedia.
   http://www.wikipedia.org/wiki/Integrated_Development_Environment
2. Peter Flynn (pflynn@ucc.ie), "The XML FAQ", Originally maintained on behalf of the World Wide Web Consortium's XML Special Interest Group, . 3.01 (2003-01-14).
   http://www.ucc.ie:8080/cocoon/xmlfaq#acr
3. "What Are Web Services?". .NET Home, May 15, 2003.
   http://www.microsoft.com/net/basics/webservices.asp
4. "Defining the Basic Elements of .NET". .NET Home, January 24, 2003.
   http://www.microsoft.com/net/basics/what asp
5. Ian Griffiths & Matthew Adams, *.NET Windows Forms in a Nutshell*, O'Reilly, 2003. (3-26)
6. Ian Griffiths, Jon Flanders, & Chris Sells, *Mastering Visual Studio. NET*, O'Reilly, 2003. (1-36)
7. Matt Culbreth, "Using Object Databases in .NET", 15 seconds Pioneering Active Server, 2003.
   http://www.15seconds.com/issue/030407.htm
8. Jeff Hadfield, group publisher, *Visual Studio Magazine* and *.NET Magazine, "VS.NET Drives XML Web Services, XML Web services promise to have the greatest* impact and most rapid adoption of any previous technology", Tech•Ed, Fawcette Technical ublications, April 11, 2002.
   http://www.ftponline.com/reports/tech-ed/0 1102/rudder/default.asp
9. Visual Studio Magazine.
   http://www.ftponline.com/vsm
10. The Microsoft .NET Framework Community.
    http://www.gotdotnet.com
11. Microsoft Corporation, "About the Common Language Runtime (CLR)", GotDotNet: The Microsoft .NET Framework Community, 2003.
    http://www.gotdotnet.com/team/clr/about_clr.aspx
12. Microsoft .NET framework, Windows ® Forms.
    http://www.windowsForms.NET
13. White Papers, Windows ® Forms, Microsoft .NET framework.
    http://www.windowsForms.NET/whitepaper/whywindowsForms.aspx
14. Shawn Burke, "Using the Microsoft .NET Framework to Create Windows-based Applications", Microsoft Corporation, December 2001.
    http://msdn.microsoft.com/netframework/using/Building/windows/default.aspx?pull=/library/en-us/dndotnet/html/netwinForms.asp#netwinForms,
15. *GDI+*, Graphics and Multimedia, MSDN library, MSDN Home.
    http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/gdiplus/gdiplus.asp

16. George Shepherd, "Windows Forms FAQ", 2001-02.
http://www.syncfusion.com/FAQ/WinForms
17. Microsoft .NET framework, Windows ® Forms.
http://www.windowsForms.com
18. *Introducing Windows Forms*, Microsoft.
http://www.developerfusion.com/show/26
19. Windows Center, Computer World.
http://www.computerworld.com/softwaretopics/os/windows
20. *The Future of .NET and XML Web Services*, Thomas Lewis & William Oellermann, Microsoft Quarterly Technical Briefing. http://www.microsoft.com/usa/presentations/QTB_ClosingSession.ppt
21. A Guide to Building Enterprise Applications on the .NET Framework. Microsoft, September 2003.
http://msdn.microsoft.com/architecture/application/default.aspx?pull=/library/en-us/dnbda/html/guidenetapp.aspx
22. Articles & White-papers, Dunstan Thomas Consultants.
http://consulting.dthomas.co.uk/software_architecture_consulting/articles_resources.htm
23. Dan Brown, ". NET Uncovered", Dunstan Thomas Consulting
http://consulting.dthomas.co.uk/ooad_articles_resources/DOTNET_Uncovered.pdf
24. .NET XML Web Services Repertory.
http://www.xmlwebservices.cc
25. *Frequently Asked Questions*. Microsoft, September 13, 2003.
http://www.microsoft.com/net/basics/faq.asp

Chapter 3: Manufacturing Execution Systems

3.1. Evolution of Manufacturing Systems

The word manufacture is derived from two Latin words "manus" (hand) and 'factus" (make); the combination of which means "made by hand" [1]. Most of modern manufacturing, though, is done using automated processes and machines and computer controls that are manually supervised. Manufacturing can be defined as an activity that integrates several components and transforms them into more value-added entity. Similarly, a manufacturing system can be defined as an integration of machines, robots and human resources that perform one or more operations to transform raw material into a final product.

Through the years, manufacturing systems have greatly evolved in to sophisticated systems that efficiently integrate a large number of resources. Computer control in manufacturing has also grown in importance and involvement by a great degree. Manufacturing has gradually evolved from NC systems to PLC's and now reaches levels of complete automation without much manual input. The goal of any manufacturing system is better productivity, reduced cycle times, less WIP, and finally, customer satisfaction. Various technologies and concepts have been used through the years for achieving these goals, some techniques are better suited for one goal while some provide better utility for another. Some concepts like flexible manufacturing systems have gained more importance than others. A flexible manufacturing system, or FMS, is based on group technology (GT). It consists of a group of processing stations (CNC machine tools) interconnected by a material handling system and a storage system. This entire set-up is controlled by an integrated computer system [1]. The name contains the term "flexible" as this system is capable of producing a variety of parts depending on the NC program fed into the different workstations. When this system consists of just a few machines (three or less), it is also referred to as a "flexible manufacturing cell" or FMC. The flexible manufacturing system consists of three components that are mentioned below:

1. **Hardware components** – These are the machines and material handling systems etc.
2. **Software components** – These are the components associated with the computer control like the CNC part programs.
3. **Human Labor** – An operator sometimes does some functions such as loading and unloading.

Manufacturing Support Systems

The manufacturing support systems are procedures and systems used by a firm to manage production and solve the technical and logistics problems associated with designing the products, planning the processes, ordering materials, controlling work-in-process as it moves through the plant, and delivering products to the customers [2]. Several of these systems are computer-based and hence there exists concepts like Computer-Integrated Manufacturing or CIM. For product and process planning, techniques like MRP and MRP II were devised. MRP or Material requirements planning was a technique used to convert the master production schedule into the individual elements of the schedule for the raw materials and components used to make the final product. MRP thus dictates the quantity of

each of the elements of the raw material and also when these components should be ordered and delivered to satisfy the production schedule. This facilitates a very minimal inventory level. MRP II or Manufacturing resource planning evolved from the MRP due to the need of tying in more efficiency with production planning. MRP II encompasses a bigger picture than what MRP does. It includes not just the materials but also the machines, resources and processes required in order to meet the schedule. MRP II is also a closed loop system, and it integrates other business aspects such as finance and forecasting into the basic MRP model. The closed loop feature of MRP II allows for a corrective action in case the feedback demands for one, thus improving the overall product.

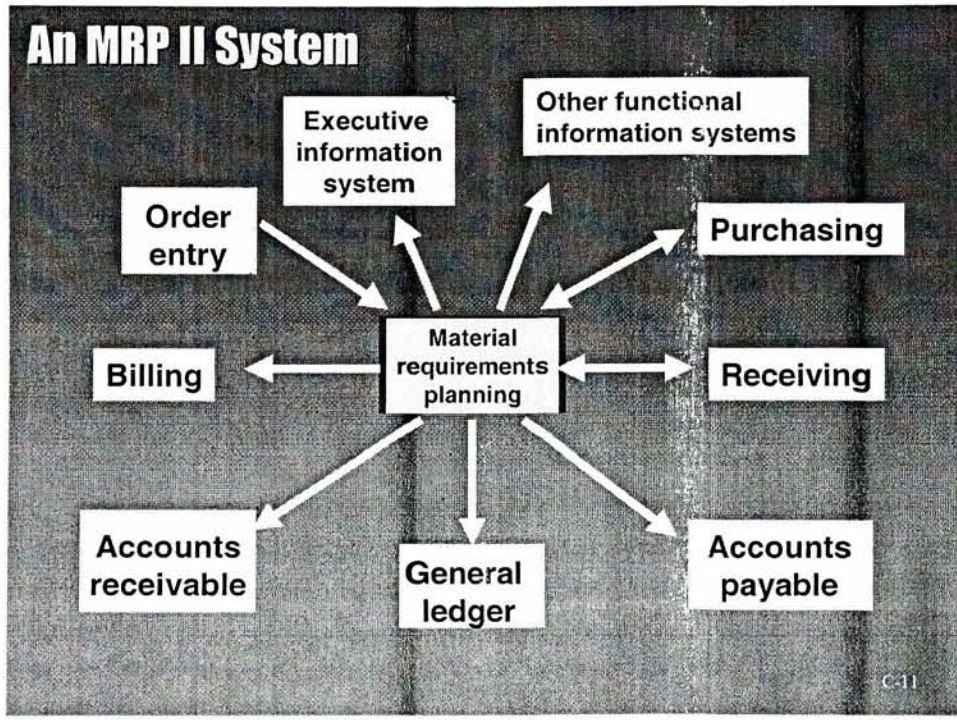

Figure 3.1 – Manufacturing Resource Planning [3]

3.2. Enterprise Resource Planning

Enterprise Resource Planning or ERP can be considered as a software system that attempts to integrate all the departments and functions across an enterprise and satisfies the information requirements of all those departments [4]. Essentially, ERP encompasses all business functions and optimizes them by integrating the business processes with technology. ERP serves as a management tool for the business management layer and provides all the required information ranging from the real-time analysis to the enterprise wide planning and control [5]. A business management layer can be considered as that layer in the enterprise that is responsible for setting the goals. It is inclined towards the strategic and tactical management of the enterprise.

American Production and Inventory Control Society (APICS, 2001) has defined ERP systems as "a method for the effective planning and controlling of all the resources needed to take, make, ship and account for customer orders in a manufacturing, distribution or service company" [6]. Figure 2 illustrates the concept of an ERP system.

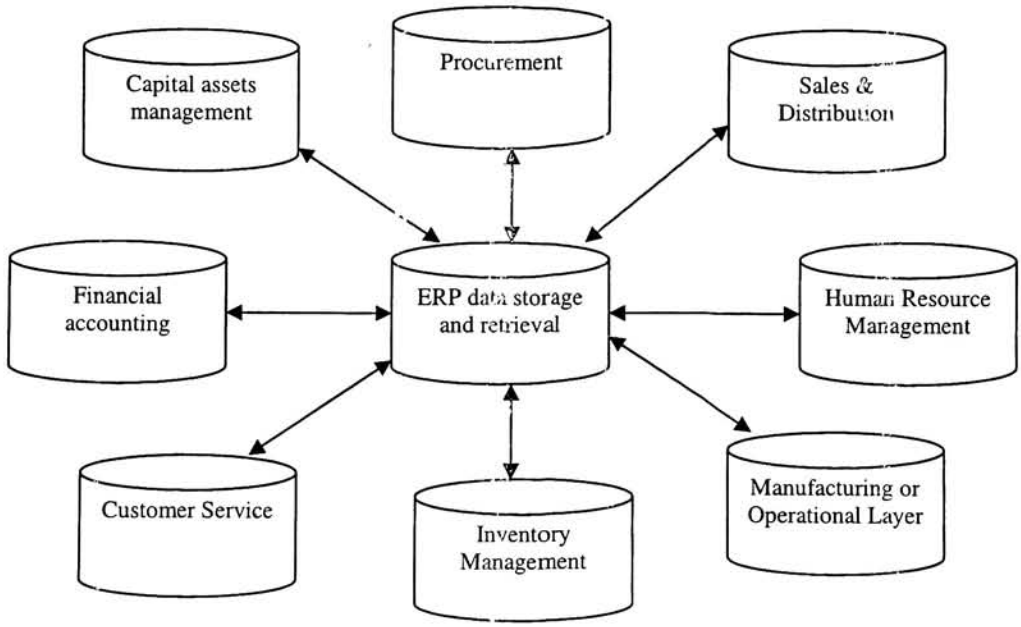

Figure 3.2 – The concept of an ERP system [5, 6]

Thus, ERP attempts to integrate the customers and the suppliers with the whole manufacturing setup for optimization of all the possible resources. Hence, for an effective ERP implementation the manner in which the information exchange takes place becomes crucial.

3.3. The Control Layer

In the context of manufacturing the Control Layer is the actual production floor or the shop floor. This is the layer where the implementation actually takes place. The machines and human resources are the components of this layer with some software technologies that are used to report the daily activities on the shop floor. The business management layer is responsible for setting the goals and the control layer is responsible for achieving the goals set by the management.

This is the section of the enterprise where the production, product design, control engineering and other manufacturing-oriented departments work together to make the final products for the customer. It consists of all the control systems like SCADA, PLC, and HMI [7]. Systems such as SCADA, which stands for supervisory control and data acquisition, are intelligent plant level systems that are used to pass data and retrieve data from the machines. They are used to keep a check on whether the production is taking place as scheduled, and also to report if there is any deviation from the target or if any downtime occurred due to any error. Thus the errors can be fixed in time and the production can be revived. MESA defines a Control System as being *"responsible for measurement, monitoring, and manipulation of production, people, products, and processes within the environs of the process or shop floor"* [8].

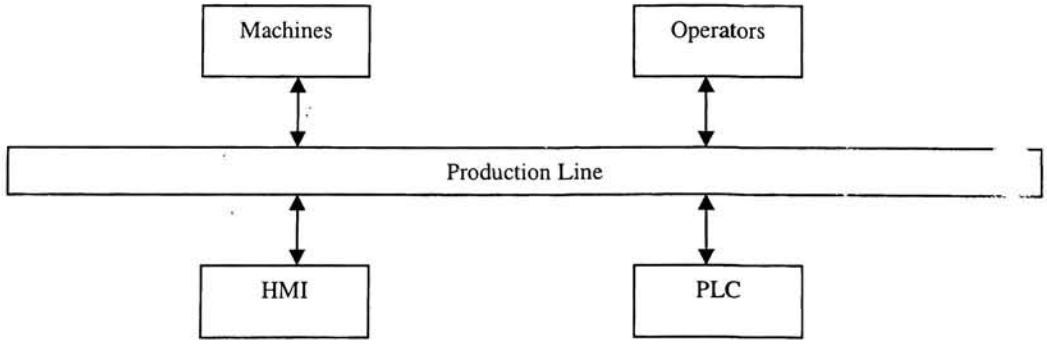

Figure 3.3 – The Control Layer [7]

Thus, there is a lot of information that is generated at this level and the different functional sections of the enterprise require that data. Control systems like SCADA are capable of generating reports at the end of a shift or a day and that report can be used as a tool for further analysis [9]. Information supplied from this layer makes the basis for making further business decisions and setting the goals for the production schedule.

3.4. A channel for making a better decision

The data collection at the operational level or the shop floor has been existent for several decades. The Business Management systems have gained popularity over the recent years. This is the reason why there is a big communication gap between these two very important layers within an industrial environment. Also, with the evolution of the World Wide Web and advancements in technology, the end users have become savvy and the demands from an industry have increased in size and diversity. The end users include the process engineers, plant supervisors, and management staff. Also, the time between the order placement and order shipment has become small, as the years have gone by. Today's customers expect nothing but the best quality at the most reasonable price and with immediate effect [10].

The other complexity that exists is that no matter how good the process plans are, the execution does not always go according to the MPS, or master production schedule. There is always a very good possibility that the machines will breakdown or that there is operator or process inefficiency or bottlenecks. To account for all these real-time occurrences and act upon them by making the necessary changes to get the product lines flowing again, effective communication between the shop floor (where the plans are executed) and the top floor (where the plans are generated) is essential.

Thus, there is a very evident need for having effective information systems that bridge this information gap. This is where the concept of a MES becomes important. MES is responsible for controlling the operational layer that executes all the plans, and it does this by having connections with all the aspects of the shop floor and the management functions. Manufactures have started realizing that delivering results from the ERP or SCM systems that are put in place in the set-ups heavily depend upon the real-time information that a shop floor has to offer. To questions that the planners and the sales people need answers for, MES can provide the answers [11].

The information thus flows from the shop floor to the business management layer though a central layer such as a MES, where the data aggregation and manipulation is done. Depending on the analysis on that data, the management can make well-informed decisions on the product or the process. Thus, this middle-tier represents an information system that can be used as a channel for making better decisions.

The following figure (see figure 4) shows the relative position of the MES in a manufacturing industry.

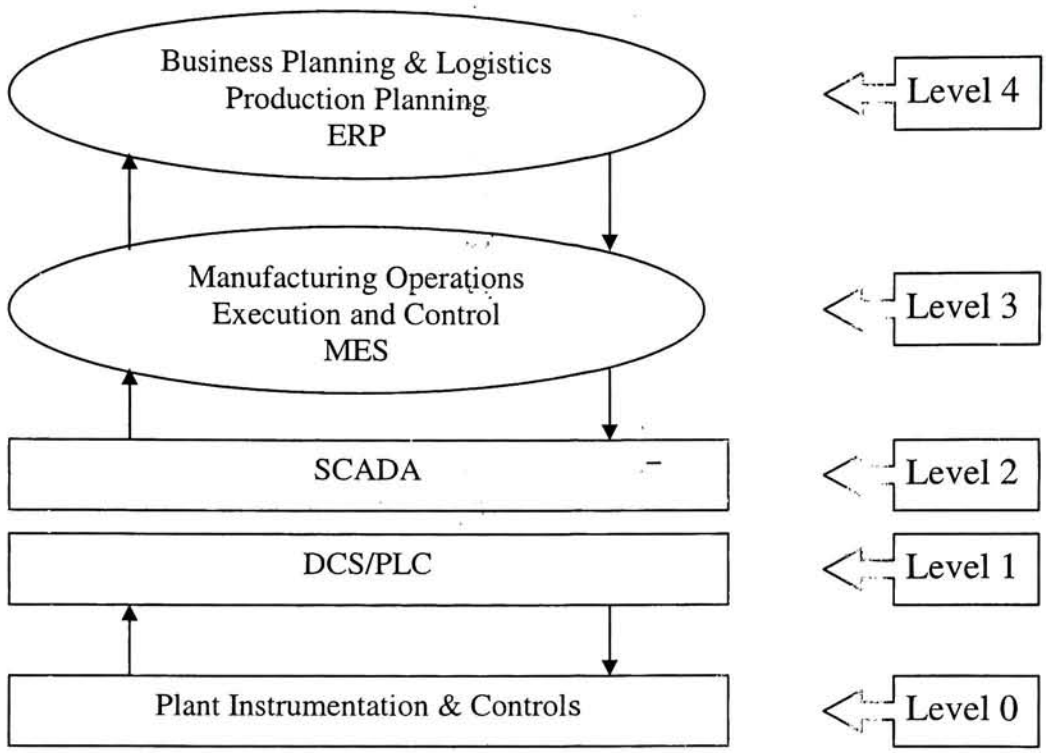

Figure 3.4 – Hierarchy of information systems in a manufacturing industry [10]

3.5. Manufacturing Execution Systems

Manufacturing Enterprise Solutions Association, or MESA, defines Manufacturing Execution Systems or MES as *"systems that deliver information enabling the optimization of production activities from order launch to finished goods. Using current and accurate data, MES guides, initiates, responds to, and reports on plant activities as they occur. The resulting rapid response to changing conditions, coupled with a focus on reducing non-value-added activities, drives effective plant operations and processes."* [8].

A MES bridges the gap between the control system at the shop floor level and the planning system at the management level through effective communication. The key is timely access to the information in order to make the desired changes and to implement the actions that are planned. A good MES architecture not only should have the capability to integrate with the existent systems, but also should be able to make space for future changes and additions to the systems. Also, time is of great importance in today's competitive world, and as the gap between the order and shipment becomes shorter by the day, timely exchange of information between the layers of a manufacturing set-up becomes crucial. Figure 5 explains the savings in time period between the various layers with a MES implementation.

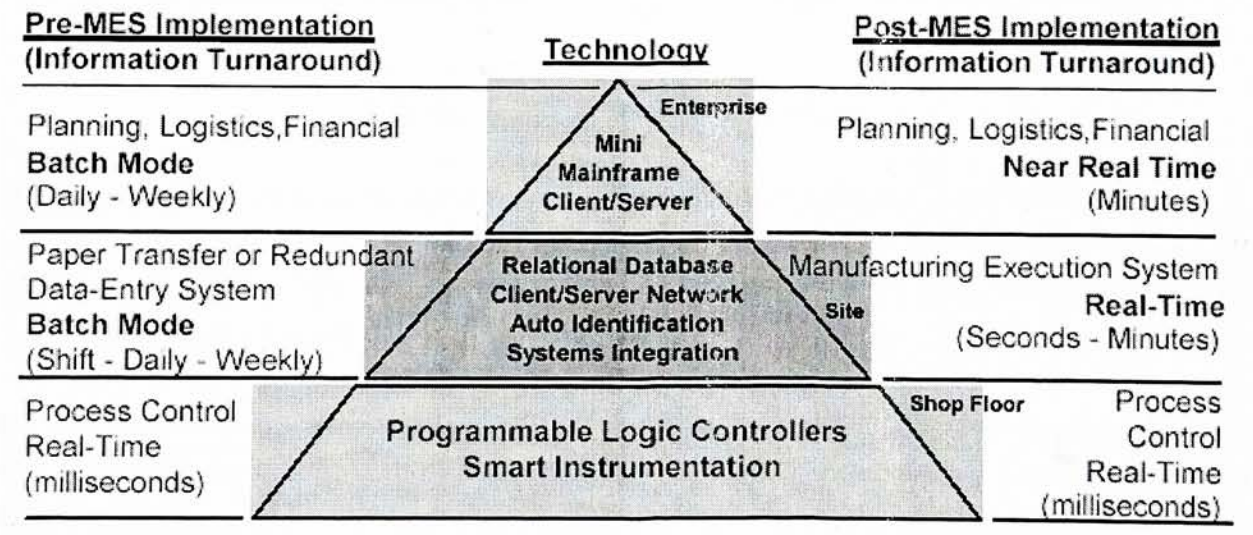

Figure 3.5 – Time benefit of a MES implementation [12]

3.5.1. MES in an Enterprise Data Flow

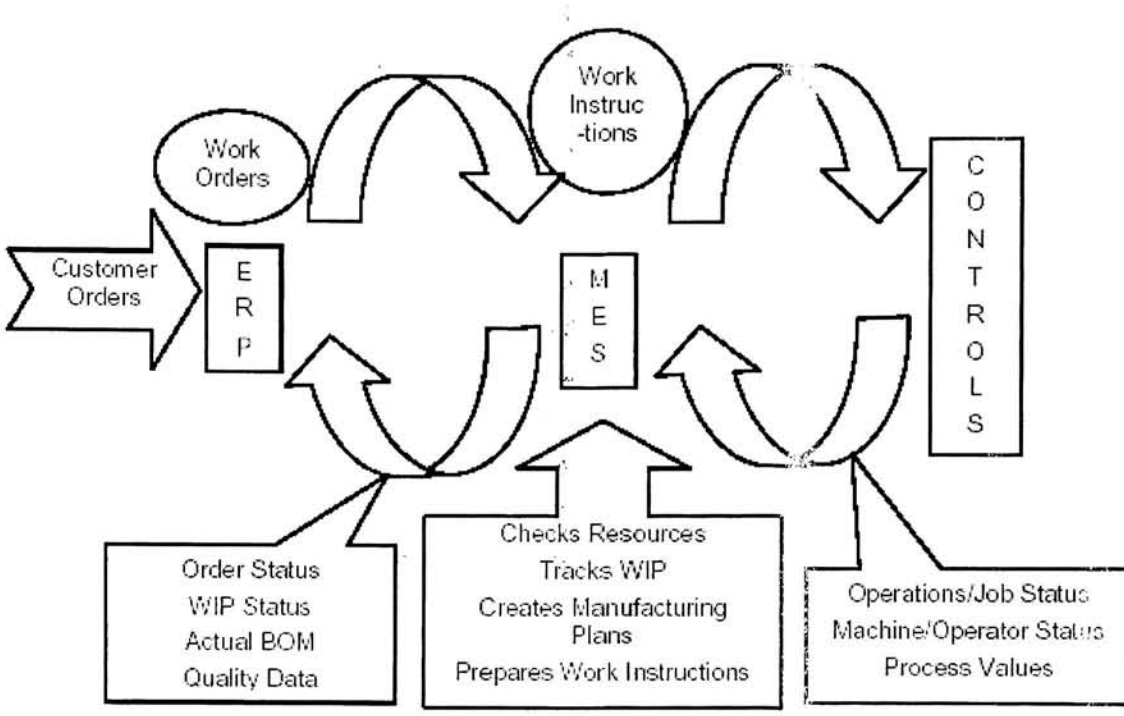

Figure 3.6 – MES, an enterprise flow model [10]

ERP systems are responsible for the customer orders, production planning, products usage, finance, and accounts and materials requirement. It does the function of sending requests to the MES layer to make the products and fill the necessary inventory to fulfill the customer orders. The time line for these ERP systems generally can span over days, weeks, months, or years [8].

The MES systems are responsible to carry out the orders received from the ERP systems. It tells the control layer what to make and how to make it. Production plans of the various products can be stored in the MES system and when required these plans can be released to the specific machines and processes in the operational or control layer to build the products. Compared to the ERP systems, the MES systems work on a time scale of days, hours, minutes or seconds depending on the situation. The various tasks that are handled by the MES are tracking the products though their manufacturing lines, storing and updating the information about the work-in-progress, or WIP, and preparing work instructions for the labor and machines to manufacture the products [8].

The control layer is the functional layer that is responsible to put all the plans into effect. Here is where the actual production takes place. The control layer uses the resources on the shop floor to fulfill the MPS or master productions schedule. Thus, it manufactures all the products necessary to fulfill all the customer orders in time for

delivery and within the desired specifications. This is the layer that works in real-time. The time factor at this level is in seconds or milliseconds. Obviously the fact is that the real world is not perfect. There are some inefficiencies and downtime issues that creep into this layer but the corrective actions have to be processed and implemented in real time too. Unscheduled changes in production, tracking the product within its sequence of operations, recognition of events that occur, are to be communicated by the control layer to the MES layer in order to get feedback from that layer for further action [8].

3.5.2. The MES context Model

The Manufacturing Execution Systems are a key element in the overall information system network of a manufacturing enterprise. MES can be considered as an information hub for the several systems shown in figure 6. These various systems have some overlap with one another as well as with the MES. Consider the example of a production plan or document control which has its place in the MES as well as in the Product and Process engineering system. The MES has to be capable of effective integration with all these systems in order for an efficient production on the shop floor. The degree of overlap may vary from industry to industry and over various implementations [8].

According to MESA, the five functional groupings that the MES should be capable to integrate with are [13, 14]:

1. **Sales and Service Management (SSM)** comprises software for sales force automation, product configurations, order management, service quoting, product returns, and post-sales service.
2. **Supply Chain Management (SCM)** includes functions such as forecasting, distribution and logistics, transportation management, electronic commerce, and advanced planning systems.
3. **Product and Process Engineering (P&PE)** includes computer aided design and manufacturing (CAD/CAM), process modeling, and product data management (PDM).
4. **Controls** are usually hybrid hardware/software systems such as distributed control systems (DCS), programmable logic controllers (PLC), distributed numerical control (DNC), supervisory control and data acquisition (SCADA) systems, and other controls designed to automate the way in which the product is being manufactured.
5. **Enterprise Resources Planning (ERP)** systems consists of those systems that provide financial, order management, production and materials planning, and related functions.

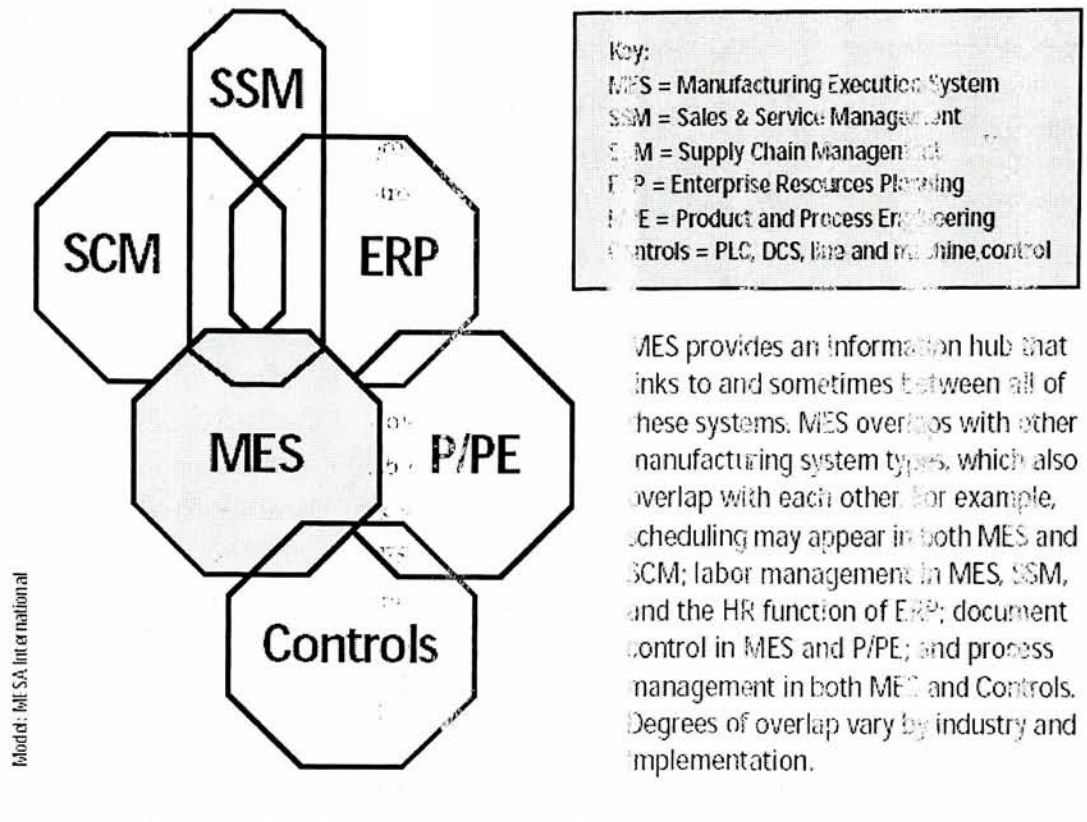

Figure 3.7 – MES context model [14]

As seen in the context model (see figure 6), MES interconnects the major five functional components of the industry. It acts as a buffer of information for these modules. The MES feeds in important information to these five core components [14].

- ERP takes the information such as cycle times, throughput and other performance oriented data from the MES.
- SCM receives data such as production line status, capacities and capabilities.
- SSM retrieves the data from the MES that is used to define the timelines for delivery, which are closely tied to events happening at the shop floor level at every given moment.
- PPE retrieves data such as quality statistics and yield from the MES.
- Controls retrieve the instructions that are actually implemented on machines from the MES that are a reflection of the optimum way that a plant should operate at any give time.

Similarly, MES also receives information from these components. Production plans stored in the MES come from the ERP system; the timings for all the activities on the production floor are driven by the data from the SCM system. Controls provide the data that is used after being analyzed in the MES.

3.5.3. The MES functional Model

MESA International has defined 11 principle functions for a MES (see Figure 7). The shop floor is a vast area of activities that take place simultaneously and often depend on the information from the other occurrences on the shop floor. MES is the glue, which holds all this information together and releases it if and when deemed necessary.

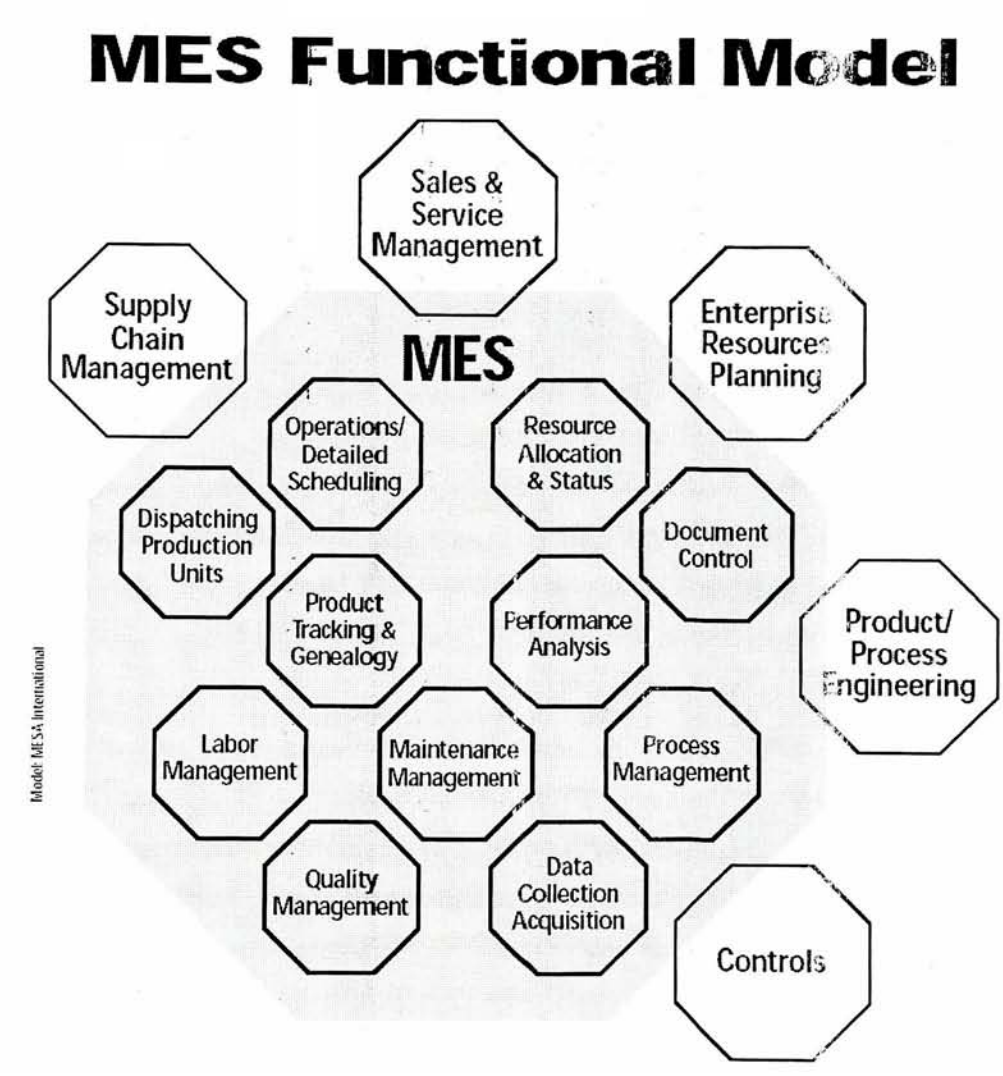

Figure 3.8 – MES functional model [14]

These 11 functionalities can be explained as follows [15]:

1. Resource allocation and status:

Manages resources including machines, human labor, tools, equipment and material. It is necessary in a way that it provides information as to what should be done and what resources should be used to do it. It also should store a brief history of all the resources and also is used to provide the status of activities in real time.

2. Operations/Detail Scheduling

Provides sequencing for the production depending upon the attributes, different priorities, or characteristics associated with different components. The sequence is done is such a way that all the resources are optimally utilized. It also calculates the exact times taken by the different processes.

3. Dispatching production lines

Manages the flow of production units in the form of jobs, batches, orders, lots or work orders. The dispatching information is also provided in the order in which the work needs to be done and it has the ability to incorporate changes as events happen on the shop floor in real time.

4. Document control

Documents such as control records or forms have to be maintained with the unit along with the work instructions, recipes, drawings, standard procedures, part programs, engineering change notices, shift-to-shift records. The ability to document the information about the activities that occur as they occur and their deviation if any from the planned activities needs to be tracked. It also should include the control and integrity of the environmental, health and safety regulations and certain ISO details like corrective action procedures.

5. Data collection/acquisition

This function provides an interface to obtain the production and parametric data, which populates the forms and records which are attached to the production unit. This data can be collected at the shop floor either manually or automatically at a real time rate.

6. Labor management

This function provides information of the personnel in and an up-to-the minute-time frame. It includes the attendance and overtime records. It also may be used for allocating the personnel to different activities.

7. Quality management

This function provides real time analysis of the measurements collected from manufacturing to assure quality control. It also may provide corrective actions to solve problems that occur. It can include SPC/SQC tracking and management of the off-line inspection operations and analysis in the laboratory management system.

8. Process management

This function monitors the production and either automatically corrects or provides decision support to operators for correcting and improving the in-process activities. It may include the alarm management to make sure that all elements on the shop floor are notified of any changes that are outside the tolerance levels.

9. Maintenance management

This function tracks and directs the activities to maintain the equipment and tools to insure their availability for the process and insure the scheduling for any maintenance activities that are needed. It maintains a past history of problems that have occurred that later help to diagnose many issues.

10. Product tracking and Genealogy

This function provides a sense of where the product is at all times and its disposition on the shop floor. Information such as who is working on the product, supplier information for raw material used, lot, serial number or other attributes related to the product can be stored and retrieved.

11. Performance analysis

This function provides an up-to-the-minute report of the actual manufacturing and the results along with comparisons to past history and expected business results. Performance results figures like resource utilization, cycle time, conformance to schedule, etc.

3.5.4. The Benefits of MES

MES provides a plant-wide view of what is happening and also of what should be happening in order to fulfill company objectives. It provides most industries a wide range of operational benefits, even for companies that have other information systems in place.

The several key benefits (see figures 8 and 9) provided by a MES are listed as [10, 16, 17, 18]:

- Reduced cycle times.
- Reduced WIP.
- Reduced paperwork.
- Reduced lead-time.
- Reduced product defects.
- More on-time deliveries
- Better deliveries.
- Higher returns on assets.
- Improved product quality.
- Rapid process upgrades.
- Reduced data entry time.
- Informed decision support.
- Productive and empowered employees.
- Real time production reporting.
- Enforce Regulatory conformance.
- Reduced product liability.
- Reduced capital expenses.
- Synchronization with demand.
- Reduced floor space.

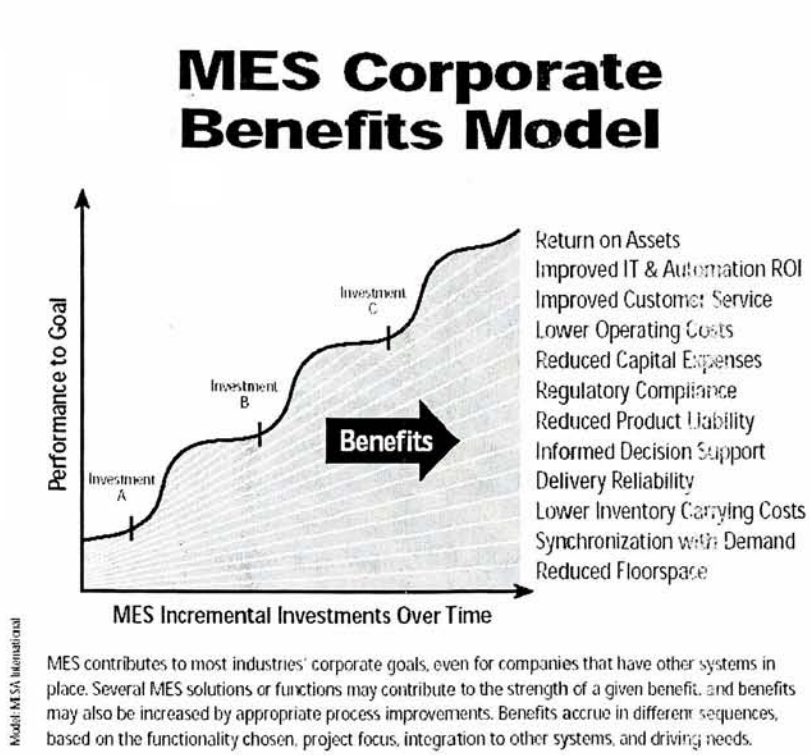

Figure 3.9 – The Corporate Benefits of MES [14]

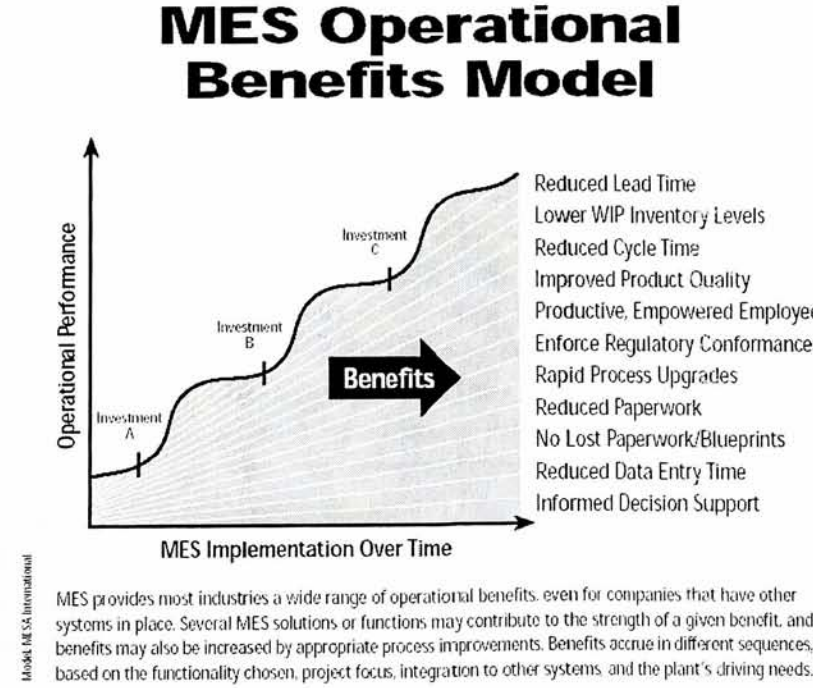

Figure 3.10 – The Operational Benefits of MES [14]

Reference:

1. Mikell P.Groover, *Fundamentals of Modern Manufacturing, Second Editio...* ohn Wiley & Sons, Inc., 2... (1-10, 860-922)
2. Mikell P.Groover, *Automation, Production Systems and Computer-Integrate... anufacturing, Second Editi...* Prentice Hall, 2001. (375-513, 753-831)
3. Raymond McLeod, Jr. and George Schell, Manufacturing Informatio... Systems, Prentice Hall, 2001 cited from Steven G.Westlund, *Appendix C, Enterprise Information Systems...* ashington University in St. ...s, Fall 2003.
   http://students.cec.wustl.edu/~t§1211b/appc.ppt
4. Christopher Koch, "*The ABCs of ERP*", cio.com, CXO media Inc. 7th March ...2.
   http://www.cio.com/research/erp/edit/erpbasics.html
5. *Gary S Rushin, Joan Clark-Rushin. , "Enterprise Resource P...nning, An Ove...w*", **Infotech Update.** New ...k: Mar/Apr 1998. Vol. 7, Iss. 2
   http://proquest.umi.com/pqdweb?index=8&sid=1&srchmode=1&vinst=PRO... fmt=6&startpage=-1&clientid=3589&vname=PQD&did=000000032052883&ts=1074647218&v... e=PQD&rqt=309&TS=107...7232&clientId=3589
6. Fiona Fui-Hoon Nah, "*Enterprise resource planning Solutions and Manage...*", Idea group publishing, 20... (Chp.3)
7. Solutions, Siemens Automation and Drives.
   http://www.siemens-industry.co.uk/automation-solutions/industrial.asp
8. "*Controls Definition & MES to Controls Data Flow Possibilities*", White Pa... Number 3, MESA Internati... February 2000.
   http://www.mesa.org/education_center/content.asp?id=11&type=wp
9. "SCADA Factory and Lab Automation Software", High Tech Services, 2003.
   http://www.htservices.com/Tools/Scada/
10. "*Manufacturing Execution Systems*", A Concept Note, TATA consultancy se...ces, Mumbai, India Februa... 002.
    http://www.tcs.com/0_industry_practices/manufacturing/downloads/MES%20...mer.pdf
11. Doug Bartholomew, "*MES revisited*", industryweek.com, Penton Media Inc. ...rch 16, 1998
    http://www.industryweek.com/CurrentArticles/asp/articles.asp?ArticleID=24...
12. "MES Benefits", White paper series, Interwave technology, Rockwell Automa...n, Inc. 2001.
    http://www.interwavetech.com/pdfs/ITI_WP_MES_Benefits.pdf
13. "*Manufacturing Execution Systems (MES)*", Manufacturing Domain Task force RFI-3, Object Management Group, November 6, 1998.
    http://xml.coverpages.org/OMG-MES971101.pdf
14. "MES Explained: A High Level Vision", White Paper Number 6, MESA International, September 1997.
    http://www.mesa.org/education_center/content.asp?id=6&type=wp
15. "MES Functionalities & MRP to MES Data Flow Possibilities", White Paper Number 2, MESA International March 1997.
    http://www.mesa.org/education_center/content.asp?id=8&type=wp
16. "The Benefits of MES: A repot from the field", White Paper Number 1, MESA International, May 1997.
    http://www.mesa.org/education_center/content.asp?id=7&type=wp

17. Lawrence S. Gould, *"MES: Will e-commerce drive it into Auto?"*, Feature article, automotive design and production, September 2001.
http://www.autofieldguide.com/articles/010008.html
18. Jonathan Kall, "Manufacturing Execution Systems: Leveraging data for competitive advantage", Quality Digest, August 1999.
http://www.qualitydigest.com/aug99/html/body_mes.html

Chapter 4: Work done in the CAMCELL

4.1 The CAMCELL [1]

This study aims at development of a MES and will implement some of its functionalities. In order to implement the functionalities, data will be required. This data will represent the real-time information that is received from the shop floor of an industry environment in the real world. The CAMCELL is a FMC that simulates such a shop floor environment, and is the source for the data collection from the control layer for the implementation of this study. The CAMCELL was an integrated manufacturing facility that evolved over several years at the department of Industrial & Systems Engineering at Rochester Institute of Technology. It had a completely automated and operational production facility. Previous research carried out in the Advanced Systems Integration Lab (ASI Lab) had the privilege of having a completely operational and functional CAMCELL. These studies focused on methods to collect the data from the control layer of the CAMCELL using the Siemens PC based automation technology. Thus, the data that was collected was all in real-time and was used for the further enhancements and analysis. This study will use this data (real-time information) collected from the previous studies and use it as an effective tool for the development and implementation of a MES.

CAMCELL can be best described from three different points of view:

1. Manufacturing & Material Handling

CAMCELL consisted of two CNC machining centers (a Mill and a Lathe), a vision/inspection station and a load/unload dock. Two closed loop conveyor systems had eight main functional stations interconnected these stations (along with four assembly stations) in the facility. The two conveyors were placed end-to-end in an "L-shape" configuration (see Figure 4.1). The conveyors were the material handling system and a buffer for up to twenty 8' x 8' size pallets, which had the capability to hold the fixtures and material needed for the CAMCELL. The pallets were tracked using radio frequency tags, which could store up to 16 bytes of data.

2. Product/Process Flow

The CAMCELL had the capability to manufacture any product that could be made using a mill or a lathe or a combination of both the CNC machining centers. Conceptually, it represented a type of a FMS, which had the flexibility of producing various types of parts. The process flow is as follows: the material and tools required were loaded at the dock, these then would travel to the CNC machining centers as per their requirement using the conveyor. Then they would go to the inspection station and then finally return to the dock for unloading.

4. Computer Hardware architecture

The CAMCELL hardware architecture evolved into a system that implemented the five-level hierarchy in manufacturing automation as defined by the National Institute of Standards and Technology or NIST. The five levels in the CAMCELL architecture are: the tool level, the station level, the cell level, the center level and the plant level (see Figure 4.2).

5. Information Flow through the CAMCELL

Information flow through the CAMCELL can be explained as five logical modules (see Figure 4.3):

5.1 Order Manipulation – The ORDPRO program performs the function of receiving, validating, processing and handling of customer orders. This feeds information needed by the Sales and Service Management module within the enterprise and maps the CAMCELL into the MES context.

5.2 Scheduling – The scheduler task program supplies information to the Process / Process Engineering. The scheduler is responsible for setting the order of production and other process details. The scheduler program and the router program are responsible for making the products using the available resources.

5.3 Routing – The ROUTER program is responsible for sequencing all the steps within the CAMCELL. It is located at the server end and assigns the orders to each of the stations.

5.4 Station Controller – The CAMCELL has eight stations, and each of these performs a specific task. The ROUTER communicates with each of these station tasks like the LATHE task, ROBOT task and the TERCO task etc. Information is interacting with the '. SQ' files.

5.5 Support – These involve the 'Inquire' module and the individual pallet controller. They are responsible for the information at the tool level of the CAMCELL where they can get information regarding the sensors and release or stop the pallets etc.

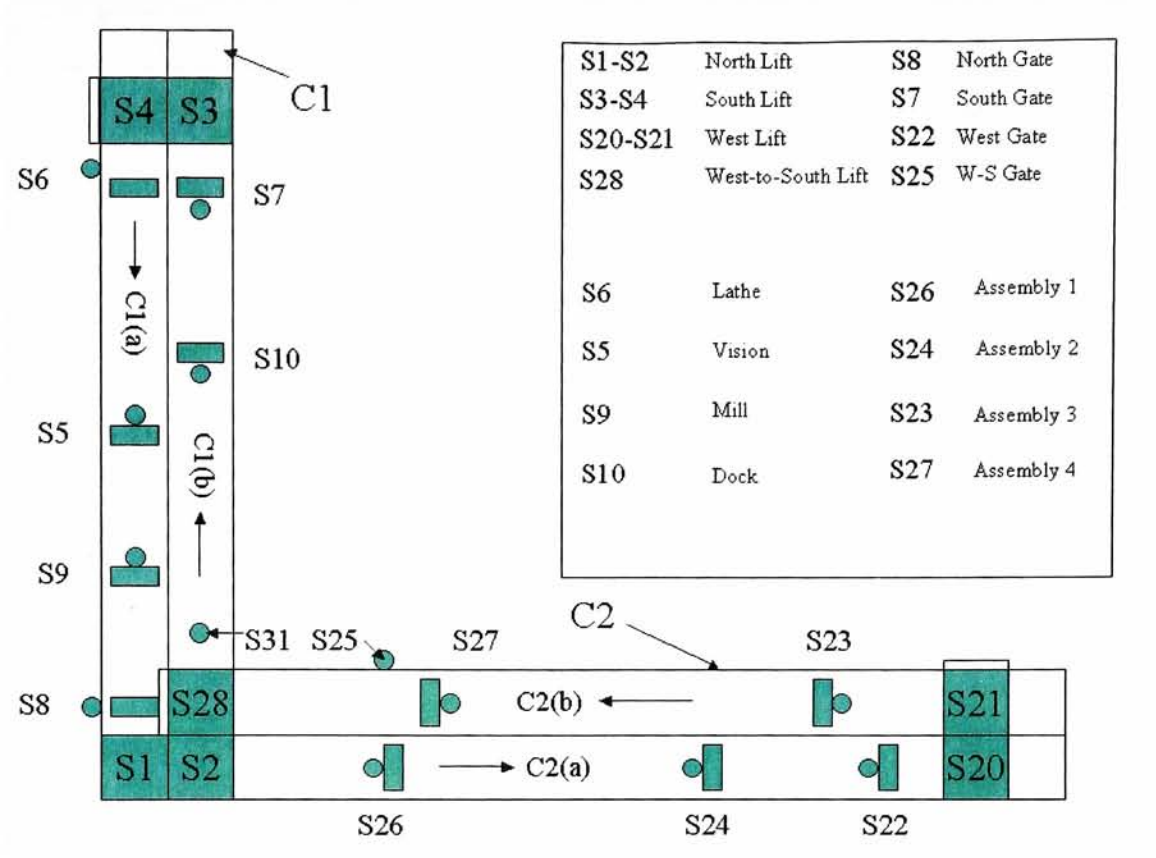

Figure 4.1 – Schematic of the CAMCELL material handling system [1]

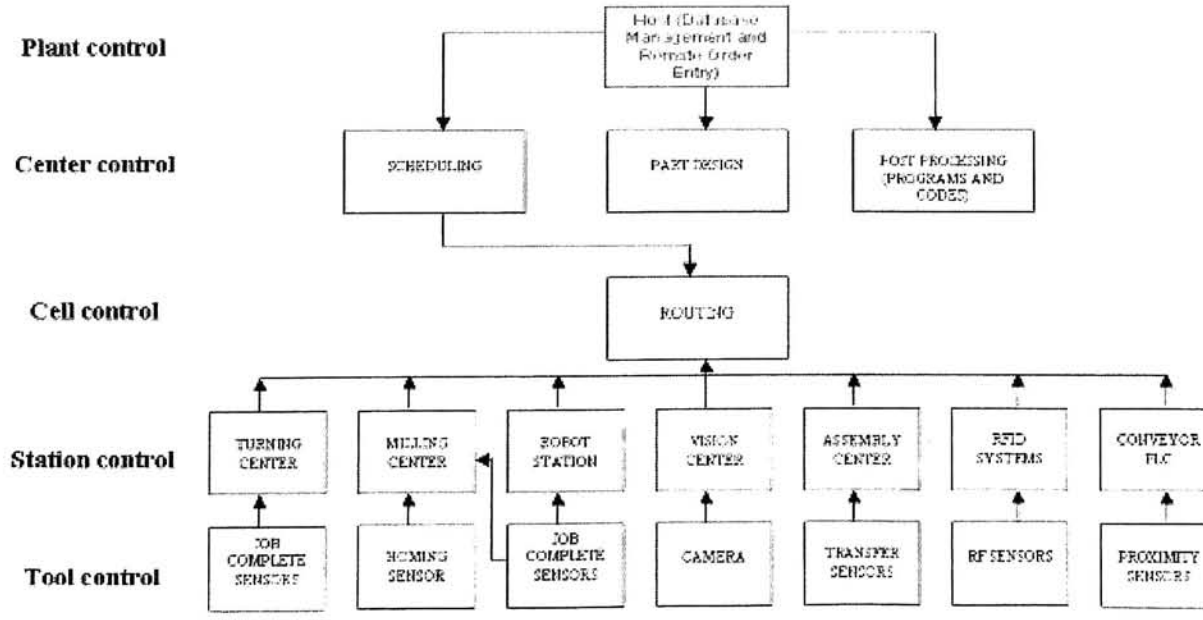

Figure 4.2 – The CAMCELL hardware architecture [1]

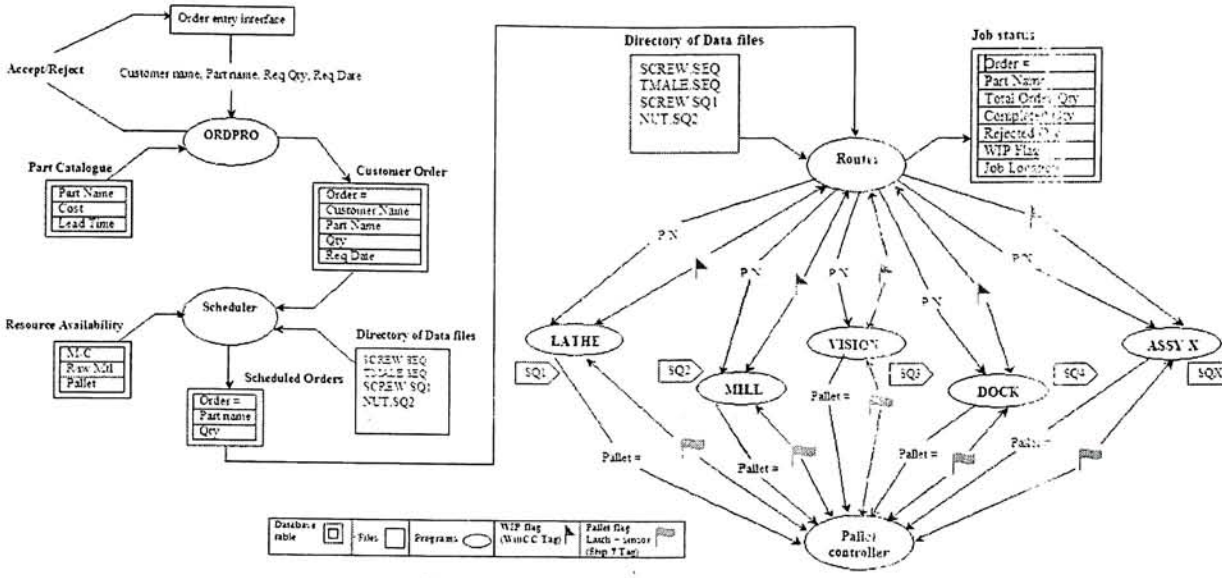

Figure 4.3 – Data Flow for the CAMCELL [2]

4.2 Systems Integration using Siemens PC based automation technology [1]

This research paper was attempts to understand the Siemens' PC based automation technology and implement it by developing a manufacturing execution system to control a FMC called CAMCELL. It concentrates on systems integration, which combine various different components of a system to merge their functional and technical characteristics into an interoperable unit. It looks at the application development environment offered by Siemens called Step 7 and WinCC. The data transfer standard called Profibus or Process Field Bus was used with the CAMCELL application. Graphical programming languages like ladder logic and functional block diagrams were used to define the logic for the system. This paper also provides support for the Siemens PC based automation technology for their claim to provide open architecture for developing applications.

4.3 A Manufacturing Execution System using Siemens' PC Based Automation Technology [2]

This research facilitated the development of a manufacturing execution system to control a FMS using Siemens PC based automation technology and Microsoft's' database technology. The paper reviews the three layers of the CIM model: the SQL and Normalization, Client/Server methodology for data management systems, and ODBC for CIM database for real time data. Using the WinCC provided by Siemens, applications were developed for the CAMCELL, which connected to FoxPro databases using ADO and remote view. These objects used ODBC and DSN to connect to the database. The research provides evidence that WinCC and FoxPro were well suited for development of MES to control the CAMCELL. WinCC was used as a SCADA HMI and achieved the plant floor integration through its open connectivity architecture. This paper also discusses the software architecture and the information flow in a CAMCELL using Data Flow Diagrams or DFD. It proves the ability of WinCC to capture the real time data from the control layer into the MES database (Visual FoxPro). It also proposes various interface screens for controlling and monitoring the CAMCELL.

Reference:

1. Adwait Palsule, *"Systems Integration using Siemens PC based automation technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2002.
2. Prakash Gandhi, *"A Manufacturing Execution System using Siemens' PC based Automation Technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2004.

Chapter 5: Data Access Standards

5.1. Introduction to Data Access

Over the past 20 years, databases and database applications have gained a lot of importance in the business world. Database applications are the cornerstone of any and every enterprise today and information located in these databases is almost priceless. Knowingly or unknowingly each and every person in today's industry interacts with databases in one form or another. This leads into an important definition for a database and understanding of "what a database application consists of?"

A database can be described as an organized collection of information and information can be described as useful data. A database can be looked upon as an electronic filing system that enables efficient retrieval of information. Each database is made up of one or more tables. These tables store the required data; each table is made up of multiple rows and columns. An individual row is called a 'record' that stores information relating to an entity, and each record has several attributes defined for that 'entity' that are stored or represented by the information in the respective columns. A Database application can be defined as "*An application program or set of related programs that is used to perform a series of activities on behalf of the database users*" [1]. Every database application performs some combination of four basic operations. These basic operations are:

1. **Create** a record into the database.
2. **Read** the data from the database and present it in the desired format to the user.
3. **Update** the database with the necessary changes.
4. **Delete** the unnecessary data from the database.

These database applications have a very wide range from individuals on their personal computers to an enterprise level ERP system.

The natural progression of these concepts is to understand what is meant by a DBMS. First and foremost a DBMS stands for '**D**ata**B**ase **M**anagement **S**ystem'.

A DBMS consists of the following components [2]:

- Software – user developed databases built on database management software like Oracle or Visual FoxPro.
- Hardware – personal computers, client machines and mainframe computers for access of the data.
- Applications – GUI components like forms and reports and other programs.
- Personnel – the end user, the developer and the administrator.

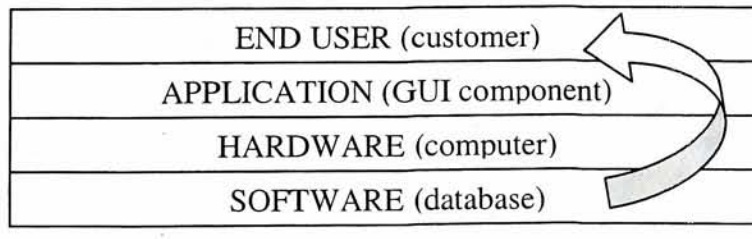

Figure 5.1 – Basic Components of a DBMS [2]

Thus, within a DBMS, the end user can retrieve the data stored in the database through some means of data flow. Thus, the means for the data flow or in other words the means for **data access** become crucial to the DBMS and have a significant impact on the design and behavior of the application.

Historically, for all database applications, companies used a single custom-made DBMS. All the databases within the DBMS and all the applications were designed using that same DBMS. These applications were thus streamlined and very application specific. However, as the technology began to grow and the world of computers and databases grew exponentially, the companies had an option of using different DBMS's. Each of the DBMS had its own advantage, some were cheaper than others, and some had better performance. Thus, a business had the opportunity to make optimum use of their capital to best suit their application. As it turned out, a company that worked with just one DBMS now had several of them. Also, with the advent of the PC, several powerful applications and tools for manipulating and dealing with data became easily available and very cheap. Thus the data was now effectively scattered across a host of these personal computers and there was a definite need for these machines to interact with different databases, which were not necessarily compatible with them. Soon, with the advent of the client-server technology the businesses took a path wherein the cheap PC's would be used as clients to provide the interactivity to manipulate the data and the mainframe computers would be the backend to support the applications and provide the necessary data to the front end. The data access between the front end and the backend thus provided an opportunity and developed a need for a very comprehensive and modular data access mode. Also, software vendors developing DBMS's were forced to write DBMS specific codes and applications to support their products (see figure 2). Thus, a large amount of the capital would be spent just for the maintenance and design of a DBMS rather than on the actual application. Thus, a need arose for a way to access different DBMS's in a very modular and an interoperable manner. This led to the development of 'data access standards'. Interoperability can be defined as the "*ability of a single application to access different database management systems through the same source code*" [3]. This interoperable data access standard needed to be independent of any single DBMS. This led to the development of the **open database connectivity**.

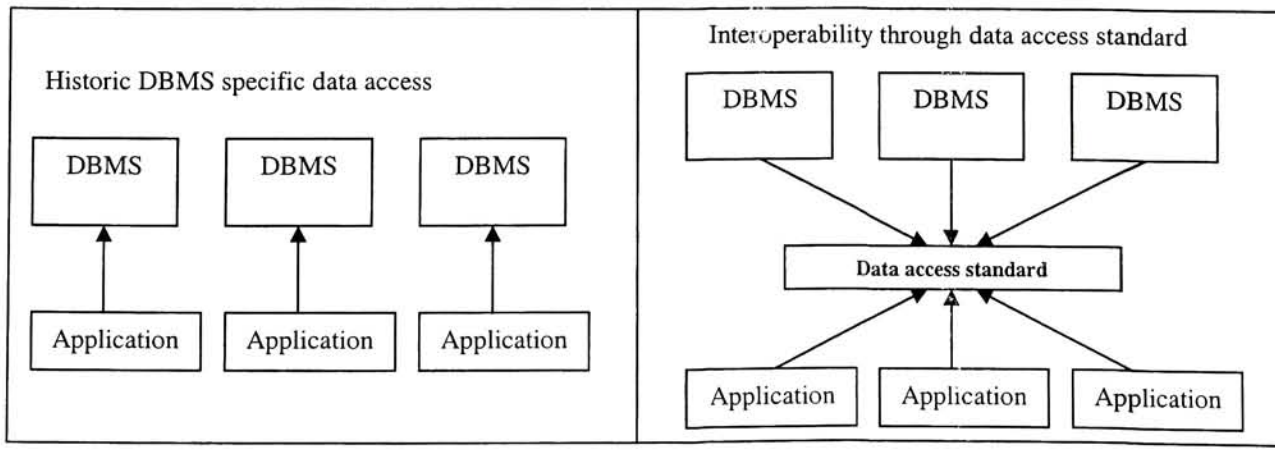

Figure 5.2 – Need for a Data Access Standard

5.2. Understanding Open DataBase Connectivity (ODBC)

ODBC is an application-programming interface (API) introduced by Microsoft that allows applications to access databases by using SQL. The ODBC API is a call level interface (CLI) [4]. Thus, in effect, an application can access remote data, using this standard, independent of the DBMS that the data resides on. ODBC is dependent upon database-specific drivers to convert the ODBC calls into the format that can interact with that specific database. Consequently, an ODBC application can be created with the purpose or goal of the application in mind and not the DBMS. This provides great flexibility to the developer for integrating the application with data that may be scattered over different DBMS's. At the time of development the programmer needs to know just the ODBC language, which is a combination of the ODBC function calls.

The ODBC architecture

The ODBC architecture [5] has four components (see figure 3):

1. Application.
2. Driver Manager.
3. Driver.
4. Data Source.

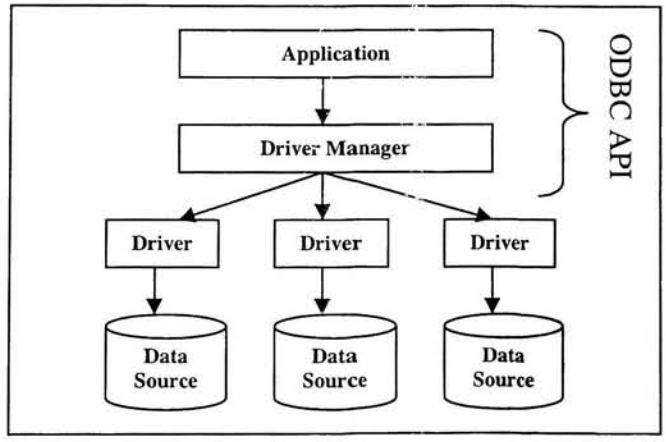

Figure 5.3 – ODBC Architecture [10]

An **application** is a program that uses standard SQL statements to load and retrieve data from the data sources through the driver manager. This is done using calls to the ODBC API. There are three types of applications, which are described next [6].

- Generic applications – These are highly interoperable applications designed to work with a variety of DBMS's.
- Vertical applications – These are applications that perform a single type of task and although they can be interoperable with several DBMS's, for a particular customer, these work with a single DBMS.
- Custom application – These are task-specific applications that generally are not interoperable and are designed for specific DBMS's and drivers.

The **Driver Manager** (for Windows, it is a DLL provided by Microsoft) is an intermediate level between the application and the drivers. After the applications make calls to the ODBC API, the driver manager determines the type of DBMS that needs to be accessed and then loads the appropriate drivers for that DBMS connection. It also processes several initialization requests, validates the parameters of the ODBC calls, and sets the target-database name to the specific driver.

The **Driver** in turn processes the ODBC calls, translates the calls into the target-database format and submits the SQL statements to the data source, receives the results and submits the results to the application. They are responsible for the correct implementation of the ODBC calls. ODBC recognizes two types of drivers: single-tier and multi-tier. A single-tier driver processes both ODBC calls and the SQL statements. The multi-tier driver processes the ODBC calls and passes the SQL statements directly to the database server [7].

The **Data Source** consists of the data that the user wants to access (the database), an appropriate supporting platform, a DBMS and the network. A data source can be anything ranging from a relational database to a text file to a spreadsheet. There are three types of data sources: User data source, System data source and File data source. User data source is one that is available to the user who created it, System data source is one that is local to a single machine, and File data source is one that can be shared by multiple database users [8, 9].

Conformance Levels

To aid the applications in determining the capabilities of the drivers and data sources, two levels of conformance have been defined: The ODBC conformance level and the SQL conformance level [9]. The ODBC conformance level determines what functions and features are available to the application from the driver. The SQL conformance level specifies the SQL statements, and data types that the driver can process.

Advantages of ODBC [10,11]

- As long as the ODBC driver for the backend system is present, the front end can access the data from any system.
- Different database systems can be accessed using the same application.
- It provides universal data access, alleviating the need to learn different programming interfaces.
- ODBC is not restricted to just the windows platform.
- ODBC works well with all databases including SQL, DB2, ORACLE, Sybase e.t.c.

Limitations of ODBC [9]

- The applications do not work as an open system as the drivers have to be purchased.
- It is limited to table-like data.
- Vendors must develop drivers for all the DBMS features and functions.

5.3. The OLE DB revolution

In 1990, Microsoft introduced its basic strategy for integrating the different data types with different application to support the **compound document**. This strategy was called **O**bject **L**inking and **E**mbedding or **OLE**. In order to understand OLE, the concept of compound document is very important. *A compound document is essentially a container of data that comes from a variety of sources such as text editors, spreadsheets, graphics, multimedia and other applications* [4]. The objective of a standard such as OLE is to provide ease of integration of a wide range of objects to form a document. OLE allows the integration of objects from one application into another application. These objects may not necessarily be organized data from databases, they can also be data located in other document that are not in a table-like manner.

As the name suggests, OLE allows data access in two ways:

Object Linking – A link of the needed object is stored in the application. Thus, the actual object is not copied, it can be accessed during runtime, and also can be accessed by multiple applications simultaneously retrieving the most current value of the object. The advantage of this is that it saves a lot of space and reduces the application size. It also provides maximum flexibility to gain access to the current data. A potential risk however is that the link might get lost and also that due to dynamic linking, the performance may suffer.

Embedding – The data of the needed object is copied into the required application. Thus, once the object is embedded, the application can access only that data. The advantage of this is that there is no risk of losing the data or the link to it and the performance is improved. The downside however is the space consumption.

OLE DB is an implementation of the OLE object standard [9]. The OLE DB objects are COM objects that support all required interfaces for these objects. Conceptually, OLE DB breaks down the DBMS functionality in to COM object, which makes the access of the data more flexible. With ODBC, the vendor needs to create drivers for all the features of the DBMS to make it operational, but with OLE DB, the vendor can create drivers for just the needed features and later add on more drivers if and when needed. It leverages the COM infrastructure, which reduces redundancy of services and provides a higher degree of interoperability not only among diverse information sources, but also among programming environments and tools already developed for this environment [12].

The OLE DB Architecture

The two basic concepts of OLE DB are consumers and providers. A **consumer** is an application or a system that utilizes the OLE DB interface. A **provider** is a system that presents the OLE DB interface to the consumer.

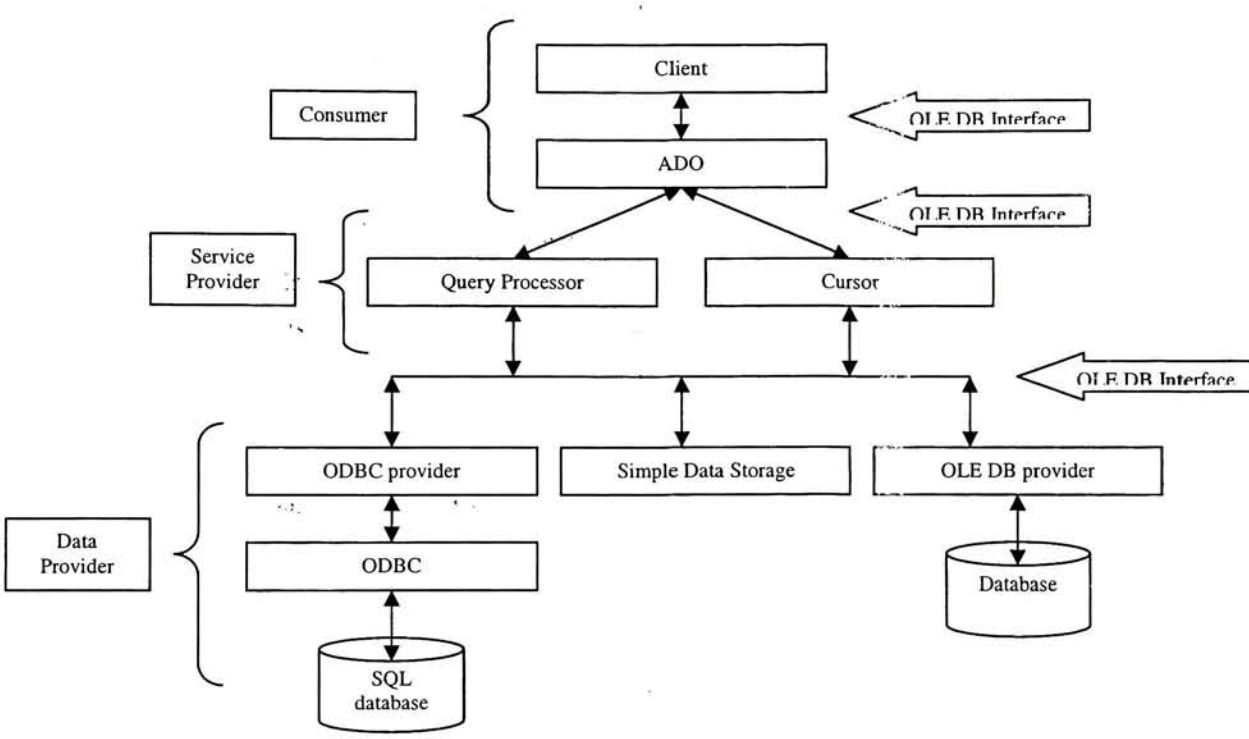

Figure 5.4 – OLE DB object model [14]

The **Data Provider** is basically a COM component with a well-defined programming interface that facilitates the data access for the OLE DB module [13, 14, 15, 16]. These data providers wrap a data source and are responsible to retrieve the data from its physical location and formatting it. Thus, a data provider does the work of exposing the data in a tabular form.

The **Service Provider** acts as a consumer and a provider. In reality, it does not own the data, but processes it before being passed on to the consumers. It can be looked upon as a transformer of data that it receives from the data provider. It fulfills one of the goals of OLE DB, which is to implement service components like a query processor and cursor engine, if and when needed.

The **Consumer** is simply an application or a tool that accesses the data from the providers.

OLE DB components interaction

Data source objects and **Data session** objects are required by the consumers to connect to the providers. Initially, the consumer creates the data source object, which in turn creates the data session object, which holds all the information like transactions, methods e.t.c. The data session objects provide methods to create the command objects and the rowsets. The Command object is responsible for preparing, specifying and executing the text commands or queries represented in the data definition language. The **Rowset** object is an abstraction that enables OLE DB to expose the data in a tabular form. All in all, the use of OLE DB can be broken down into the following components (see figure 5):

- Initialize the environment.
- Connect to a data source.
- Create and execute a command.
- Process results.
- Clean up

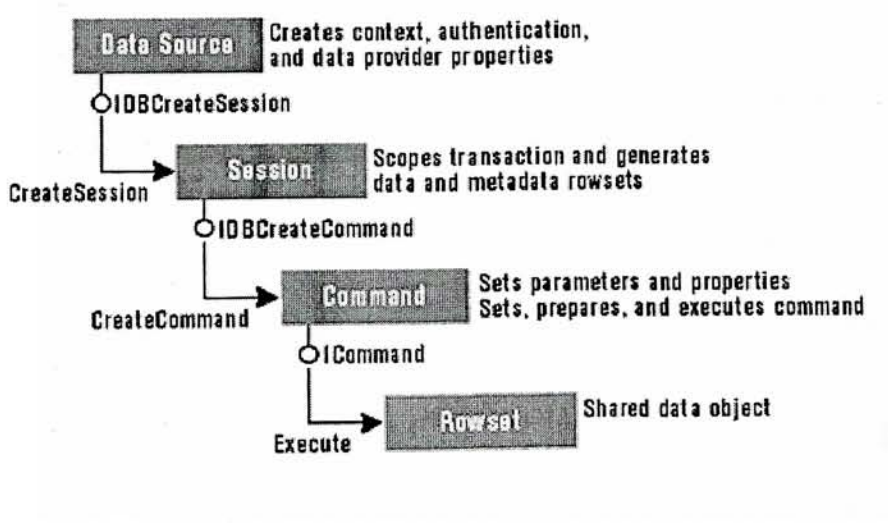

Figure 5.5 – OLE DB Component Interaction [14]

5.4. The Advent of ADO

OLE DB uses low-level calls for data access to the consumers, which makes the programming tedious and time-consuming. ADO acts as a 'wrapper' over the OLE DB layer and provides a higher-level interface to facilitate the data access. It is a simple object model that can be used by the consumers to process any OLE DB data. It can be called using several languages like VBScript, JScript, Java, C#, C++ and Visual Basic [17].

ADO stands for **A**ctiveX **D**ata **O**bjects, and provides access to Microsoft's data access technologies [17]. It acts as a consumer to the OLE DB data providers. OLE DB is a system-level interface while ADO is an application-level interface. Although, OLE DB is a very powerful tool, seldom do developers need access to the low-level details such as manually aggregating all the components, which can be provided using OLE DB. Furthermore, developers often use high-level interface tools such as Visual Basic that do not support low-level details like pointers in C++. To side-step from the low-level interaction that is done while using OLE DD, there is a much higher level programming model that acts at the application level. And this is exactly where ADO and its utility come in. Programmers can write applications over the OLE DB data using several languages.

Following are the advantages provided by ADO [18, 19]:

- Ease of Use – Data access requires very few lines of code.
- Programming language neutral – Several languages like C++, JScript, VBScript can be used to develop applications over OLE DB data.
- Provider neutral – Data can be accessed from any OLE DB source.
- No loss of OLE DB functionality – C++ programmers can access the low-level data of the OLE DB interfaces.
- Extensible – ADO can dynamically expose properties of the data provider via the collections of the data provider properties. Type extensibility is also available through access to COM components via column values.

Thus, ADO relieves the developer who is not interested in learning the intricacies of Com or OLE DB and prefers programming at a higher level using tools such as Visual Basic. Also, there is an improvement in the performance and also reduction in the development cost.

ADO Object Model

The ADO object model consists of three basic high-level components [19, 20, 21, 22]. These three components are:

- Connection Object.
- Command Object.
- Recordset Object.

These three objects can be created and destroyed independently of any other object. In addition to these three objects there are four other objects, which are error object, property object, parameter object and the field object. Each of the top-level objects does contain a property object (see figure 6)

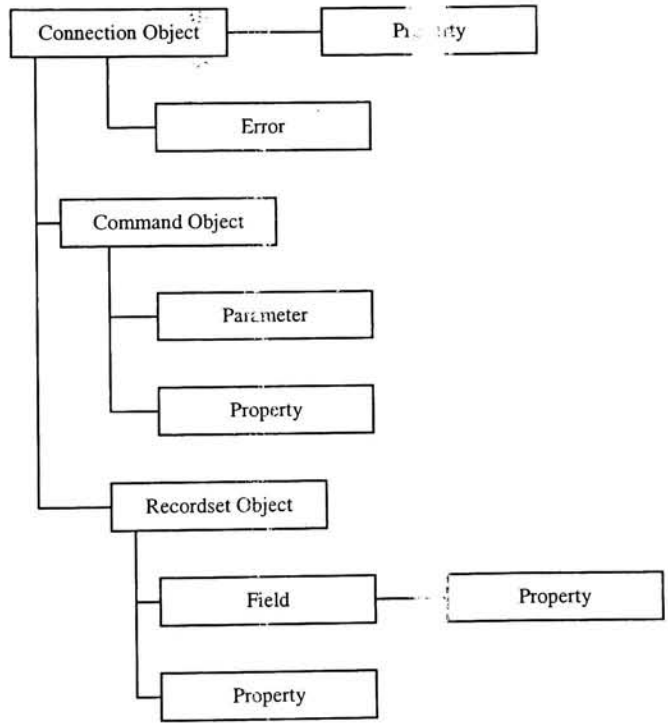

Figure 5.6 – ADO object model [19]

The **connection object** opens a connection to the data source. All ADO applications must support a connection object. This is the common gateway to all the other ADO objects.

ADO returns all the errors in a collection called the **error object**. All the ADO objects have associated **property objects** with them.

The **Command object** is used to execute the queries and stored procedures that are stored in a database. These are the SQL statements that need to be executed.

Each command object has associated parameters that are used to pass additional information and define the parameters that are required. Individual parameter information can be read or written using a unique **parameter object**.

The **Recordset object** encapsulates the OLE DB rowset object. It holds the entire set of records that are a result of executed commands. This is where the data is read and the manipulation of data is done.

The recordset object exposes a collection of **Field objects** that contain the metadata about the columns in the recordset, such as name, type, length etc.

5.5. Migration to ADO.NET

ADO.NET is a new, improved and enhanced version of ADO that was developed as apart of the Microsoft.NET initiative. It incorporates the functionality of ADO and OLE DB and has additional features that make it the data access technology for the future. In particular, it facilitates the transformation of XML documents to and from the databases, and has the unique ability to create and process in-memory pseudo databases called **'datasets'** [9].

ADO.NET is the binding force that integrates all the applications developed in the **Microsoft.NET framework** and the DBMS and databases from which the data is derived (see figure 7).

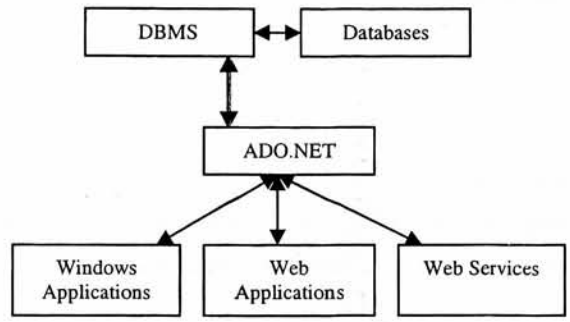

Figure 5.7 – Where ADO.NET fits in

Thus, ADO.NET is an integral part of the .NET framework. The important feature of the .NET framework is that it facilitates development of applications in various languages and then provides a feature to integrate these applications into their parent application. Thus, the interface used by the user can access all types of application independent of the languages that the individual applications are developed in. This is achieved through the **Common Language Runtime** (CLR) [23].

Extensible Markup Language (XML)

"XML is easily comprehensible to anyone who understands HTML, but it is much more powerful. More than just a markup language, XML is a metalanguage *-- a language used to define new markup languages. With XML, you can create a language crafted specifically for your application or domain".* Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML [24, 25]. XML has a clear separation between the document structure, content and materialization. With XML, the developer can create his or her own tags and elements. Any XML document has two distinct sections. The first defines the structure of the document called the 'document type declaration' or **DTD**, and the second is called the document data, which contains the actual data.

A progression from ADO

ADO.NET was developed with much of its basics similar to ADO with a few changes, both internal and external. The main internal change is that ADO.NET is based entirely upon XML. Externally, it differs from ADO in such a manner that there is no recordset in ADO.NET. The functionality of the recordset has been divided in to three parts: DataReader object, DataSet object and DataSetCommand object. By dividing the recordset functionality from the ADO into distinct element **ADO.NET provides much more flexibility, higher performance and tailored support for each element** in the application [27]. The dataset can be considered as a fully functional and independent **in-memory database**. It has all the functionality of normal databases. This is the aspect of ADO.NET, which represents a merger of two technologies: ADO and XML. The dataset data can be constructed from data in different databases, which are managed by different database management systems. Once the dataset is constructed, the contents can be formatted as a XML document and similarly a XML schema for the dataset can be generated.

ADO.NET architecture

There are two basic components of ADO.NET in order to part data access from data manipulation. These two components are: the **DataSet** and the **.NET framework data provider** (see figure 8). The .NET framework data provider consists of Connection object, Command object, DataReader object and DataAdapter object [28, 29].

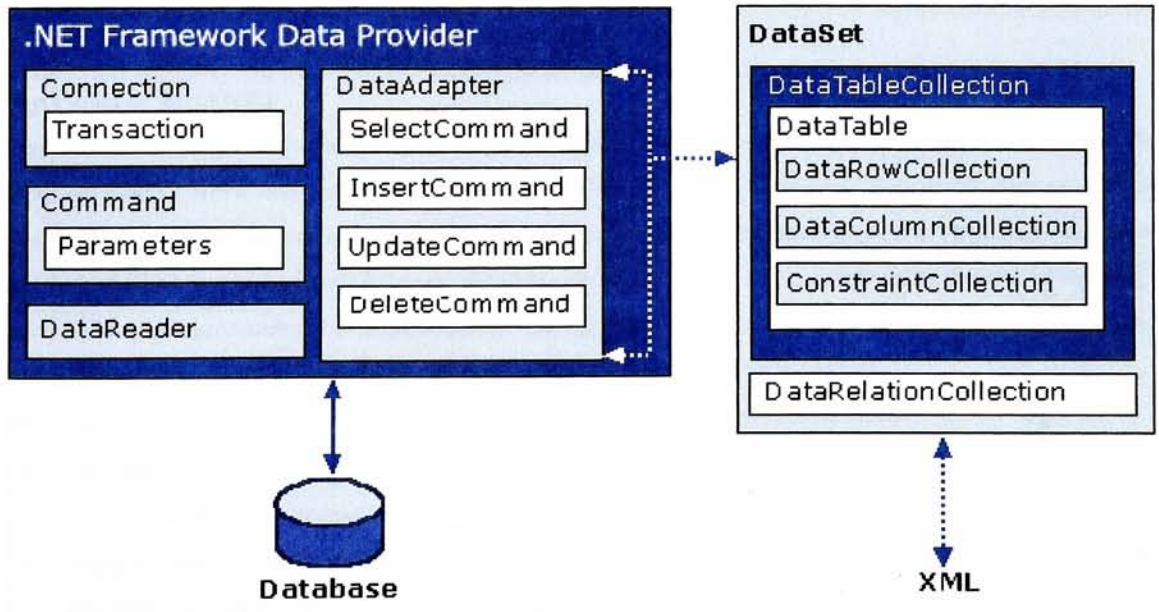

Figure 5.8 – ADO.NET architecture [28]

As mentioned earlier, a data set is an in-memory database and has the capability to support the data being disconnected from the original source. It can be used with multiple and differing data sources, used with XML data, or used to manage data local to the application. It represents a complete set of data including related tables, constraints, and relationships among the tables. The DataSet can also persist and reload its contents as XML and its schema as XML Schema definition language (XSD) schema. The DataSet has two basic components: DataTableCollection and DataRelationCollection. The DataTableCollection contains zero or more DataTable objects, which are in turn made up of three collections: Columns, Rows, and Constraints. The DataRelationCollection contains zero or more DataRelations.

A data provider is a class library that provides ADO.NET services. There are currently three Microsoft-supplied data providers: the OLE DB data provider, the SQLClient data provider, and the OracleClient data provider. But ADO.NET does have a capability of including more of them. The **Connection** object initiates and gains the connection to the data source. The **Command** object holds the SQL statements and the stored procedure that are to be executed. **DataAdapter** in turn manages the interactions between the dataset and the data source. The DataAdapter contains four Command objects: the SelectCommand, UpdateCommand, InsertCommand, and DeleteCommand. The DataAdapter uses the SelectCommand to fill a DataSet and uses the remaining three commands to transmit changes back to the data source, as required. A **DataReader** is an object for obtaining a forward-only, read-only stream of data from a data source. It can be created only by calling the ExecuteReader method of a Command.

Design Goals of ADO.NET

As application development has evolved, new applications have become loosely coupled based on the Web application model. More and more of today's applications use XML to encode data to be passed over network connections. Web applications use HTTP as the fabric for communication between tiers, and therefore must explicitly handle maintaining state between requests. This new model is very different from the connected, tightly coupled style of programming that characterized the client/server era, where a connection was held open for the duration of the program's lifetime and no special handling of state was required [30].

While developing ADO.NET, the following design goals were taken into consideration:

- **Leverage the present ADO knowledge** – The ADO functionality is not lost but it is used to step up to get a much simpler data access model.
- **Support the N-tier programming Model** – ADO.Net supports disconnected data very efficiently and provides support for the N-tier programming Model for which most of the current applications are written.
- **Integrate XML support** – XML and ADO.NET are very well integrated. The XML support is built into the ADO.NET at a very fundamental level.

A Closing Note

As the Internet and web became more interactive and widely used, the need for a middleware became more and more prominent and evident. The middleware is the logical layer in which the data access components usually reside and carry out their functions. With web technology and computer technology increasing at an exponential level, many different tools and platform were born. The introduction of .NET is the next step in Microsoft's maturing component technology [31, 32]. Each new tool has its own merits and can perform certain functions in a more efficient manner than the others. Be it Internet on a web server or Intranet on a network server, there is a need for interoperability work that is shared across heterogeneous platforms and hence a robust data access technology is required. Microsoft has optimized ADO Recordsets for data transmission in a Web environment and XML acts as an universal glue between tiers, no matter which platforms are involved. As interoperability and scalability increase, ADO does not give the best possible answer because it is not XML-based but **ADO.NET is**.

5.6. Glossary

Object

In object-oriented programming, objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. In between, each object is made into a generic class of object and even more generic classes are defined so that objects can share models and reuse the class definitions in their code. Each object is an instance of a particular class or subclass with the class's own methods or procedures and data variables. An object is what actually runs in the computer [33].

Compound document

In information technology, a compound document is an organized collection of user interfaces that form a single integrated perceptual environment. A compound document includes a data structure that contains different data types, such as text, audio files, and motion video files. A compound document is also an application environment containing program objects that can be interlinked and interacted with by a user [34].

Component Object Model

Component Object Model (COM) is Microsoft's framework for developing and supporting program component objects. It is aimed at providing a framework for the interoperation of distributed objects in a network that is supported by other major companies in the computer industry [35].

Call Level Interface

A programming interface designed to support SQL access to databases from shrink-wrapped application programs [36].

Document Type Declaration (DTD)

It is that part of an XL document where the structure of a document is defined by the developer. It consists of the various elements and their parameters to be used in the document [9].

Query Processor

Query processor is an object that accepts the SQL syntax, selects the needed command and executes the chosen plan [37].

Cursor Engine

It is the underlying technology of remote data access that is responsible for retrieving and updating the results of queries sent to the DBMS [38].

Reference:

1. Fred R. McFadden, Jeffrey A.Hoffer, Mary B.Prescott, *Modern Database Management, Fifth Edition*, Addison-Wesley, May 1999. (1-14)
2. Nilesh Shah, *Database Systems Using ORACLE*, Prentice Hall, 2002. (1-5)
3. Introduction to ODBC, ODBC programmer's reference, MSDN library, Microsoft Corporation, 1998-2003. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odch01pr.asp
4. Amjad Umar, *Object-Oriented Client/Server Internet Environments*, Prentice Hall, 1997. (217-223, 329-332)
5. ODBC Architecture, Delivering Data to the Desktop, Universities and Colleges Software Group, 2002. http://www.liv.ac.uk/middleware/html/odbcarch.html
6. Applications, ODBC Architecture, MSDN library, Microsoft Corporation, 1998-2003. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odch03pr_1.asp
7. ODBC Architecture, Open Database Connectivity without compromise, OpenLink Software, 2003. http://www.openlinksw.com/info/docs/odbcwhp/odbcobj.htm
8. ODBC: a Technical Overview, Data Access Worldwide, 2001. www.dataaccess.com/whitepapers/odbc.htm
9. David M.Kroenke, *DATABASE PROCESSING, 9th EDITION*, Pearson Prentice Hall, 2003. (425-506)
10. Implementing ODBC, Macrovision Creative Software, 1996-2003. http://www.macrovision.ie/research/eagodbc.htm
11. Why developers should use ODBC instead of native proprietary database interfaces, DataDirect Technologies, 2002. http://www.datadirect-technologies.com/products/odbc/docs/faster-than-native-final.pdf
12. Jose A.Blakeley, "OLE DB: A Component DBMS Architecture", Microsoft Corporation, 1996. http://csdl.computer.org/comp/proceedings/icde/1996/7240/00/72400203.pdf
13. Dino Esposito, *OLE DB or ODBC? Look before you leap*, SQL SERVER magazine, November 1999. http://www.sqlmag.com/Articles/Inde5.cfm?ArticleID=6197&pg=1
14. Aaron Skonnard, *Say UDA for all your Data Access Needs*, Microsoft Interactive Developer, April 1998. http://www.microsoft.com/mind/0498/uda/uda.asp
15. OLE DB, Data Access, MSDN library, Microsoft Corporation, 1998-2003. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbstartpage1.asp
16. Jose A.Blakeley, *Data Access for masses through OLE DB*, International Conference on Management of Data and Symposium on Principles of Database Systems, Proceedings of the 1996 ACM SIGMOD international conference on Management of data, 1996. (161-172) https://portal.acm.org/ft_gateway.cfm?id=233329&type=pdf&coll=portal&dl=GUIDE&CFID=15031140&CFTOKEN=83844951
17. Mahesh Chand, *What is ADO,* Mindcracker, June 25, 2000. http://www.mindcracker.com/mindcracker/c_cafe/ado/ado1.asp
18. Nitin Paranjape, Data Access Methodologies, Times Computing, March 3, 1999. http://www.timescomputing.com/19990303/ttr1.html
19. OLE DB / ADO: Making Universal Data Access a reality, Microsoft Corporation, May 1998. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuda/html/msdn_dbado.asp

20. Mike Chapple, ActiveX Data Objects (ADO), About Inc. October 3, 2002.
http://databases.about.com/library/weekly/aa031002a.htm
21. Reshmi Nair, *A Journey towards ADO.NET*, dotnetextreme.com, 2001.
http://www.dotnetextreme.com/articles/adonetJourney.asp
22. Irinia Medvinskaya, *ADO delivers high-performance data access*, Builer.com, CNET Networks, December 5, 2001.
http://builder.com.com/5100-6373-1045835.html
23. Dan Brown, *.NET uncovered*, Dunstan Thomas Consulting.
http://consulting.dthomas.co.uk/ooad_articles_resources/DOTNET_Uncovered.pdf
24. Mark Johnson, XML for the absolute beginner, Java World, April 1999.
http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html
25. Liam Quin, *Extensible Markup Language (XML)*, W3C, August 20, 2003.
http://www.w3.org/XML/
26. Rob Macdonald, *Essentials ADO.NET*, Matt Publishing Limited, 2001.
http://www.dnjonline.com/articles/essentials/iss22_essentials.html
27. William R.Vaughn, *ADO.NET and ADO examples and Best Practices for VB programmers, 2nd edition*, Apress, 2002.
28. ADO.NET architecture, Overview of ADO.NET, MSDN library, Microsoft Corporation, 1998-2003.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconadonetarchitecture.asp
29. Rebecca M.Riordan, *Microsoft ADO.NET Step by Step*, Microsoft Press, 2002.
30. Design Goals of ADO.NET, Microsoft Corporation, 2003.
http://longhorn.msdn.microsoft.com/lhsdk/ndp/cpconwhyadonet.aspx
31. Dino Esposito, *ADO.NET: A bridge to the future*, SQL SERVER magazine, Penton Media Inc, June 2001.
http://www.sqlmag.com/Articles/Inde5.cfm?ArticleID=20744&pg=2
32. Understanding the difference between ADO and ADO.NET, Technical Brief, DataDirect Technologies, 2002.
http://www.datadirect-technologies.com/products/dotnet/docs/adovsadodotnet-final.pdf
33. Object, Definitions, TechTarget, March 03, 2003.
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci212680,00.html
34. Compound Document, Definitions, TechTarget, July 23, 2001.
http://searchwin2000.techtarget.com/sDefinition/0,,sid1_gci211827,00.html
35. Component Object Model, Definition, Jupitermedia Corporation, October 16, 2003.
http://www.webopedia.com/TERM/C/Component_Object_Model.html
36. Call Level Interface, Definition, NightFlight, December 2000.
http://www.nightflight.com/foldoc-bin/foldoc.cgi?Call-Level+Interface
37. *Chapter14-Query Processor*, Resource Kits, SQL server, Microsoft TechNet, Microsoft Corporation, 2003.
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/reskit/sql7res/part9/sqc13.asp
38. *Understanding Remote data service applications*, Microsoft Corporation, 1997.
http://www.host-web.fr/iishelp/adc/docs/adcdg01_1.htm#adc_dg_arch_vtm
39. Dino Esposito, *Dive into .NET data access*, Fawcette Technical Publications, November 21, 2001.
http://www.ftponline.com/wss/2001_12/online/online_eprods/desposito/default.aspx
40. "ADO.NET overview", GotDotNet, .NET Framework Community, Microsoft Corporation, 2002.
http://samples.gotdotnet.com/quickstart/howto/doc/adoplus/ADOPlusOverview.aspx
41. Gurneet Singh, *Getting Started with ADO.NET*, DotNetExtreme, 2001.
http://www.dotnetextreme.com/articles/GetStartADO.asp

42. Cary Jensen, Data Access in ADO.NET, Borland developer network, Borland Software Corporation, July 11, 2003. http://community.borland.com/article/0,1410,30113,00.html
43. Cary Jensen, Data Storage in ADO.NET, Borland developer network, Borland Software Corporation, July 18, 2003. http://community.borland.com/article/0,1410,30237,00.html
44. Karl Moore, *ADO.NET overview*, Juitermedia Corporation. http://www.developer.com/net/vb/article.php/10926_1540311_6
45. Dimitrios Markatos, Introduction to ADO.NET, Server Side Coding, SitePoint Pty. Ltd., January 30, 2003. http://www.sitepoint.com/article/992
46. Dan Fox, Design an effective data access architecture, Fawcette Technical Publications, July, 2002. http://www.ftponline.com/wss/2002_07/magazine/features/dfox/default.aspx

Chapter 6: The MESASI database

6.1 The real-time MES database

A database that supplies information for the MES layer is essentially getting information from the control layer. MESA defines a Control System as being *"responsible for measurement, monitoring, and manipulation of production, people, products, and processes within the environs of the process or shop floor"* [1]. The real time data that was required from the CAMCELL was captured using the Siemens PC based automation technology called WinCC. Also, the other feature of this database was the information about the clients and suppliers. This information can be edited by two channels. These two channels or modules are the client module and the administrator module. The clients can log-in to the system and enter their information and also place and view the orders under their respective accounts. The administrator, on the other hand, has access to the entire database and can access entire customer information as well. Thus for the tables in the MESASI database, the information or data is being fed in from either the shop floor by data-acquisition systems or by the end-users of the MES system (see Figure x.1).

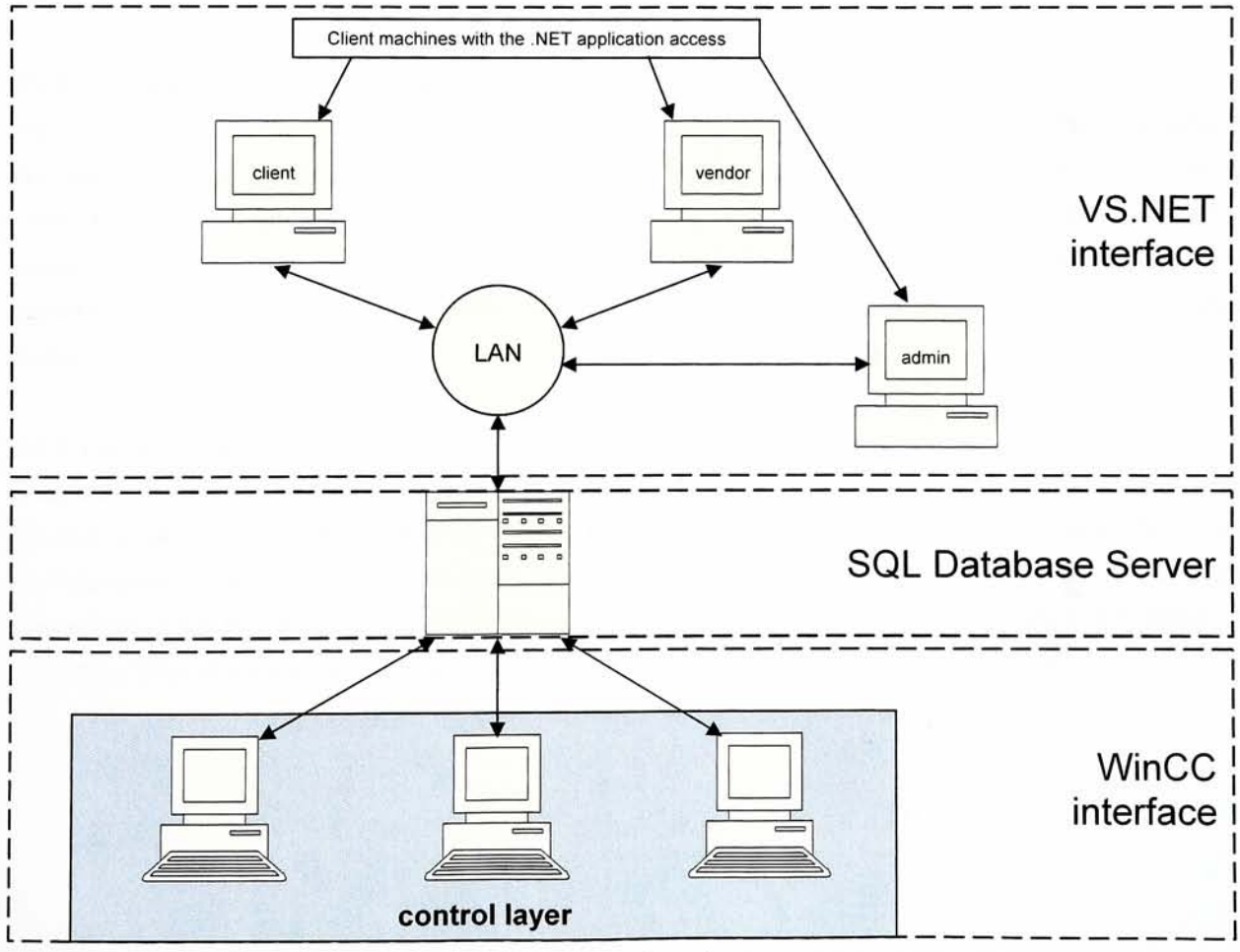

Figure 6.1 – Three tier architecture

This section provides a brief description of the various tables in the MESASI database and explains what information can be stored in the tables and how the information is important for the MES application.

6.1.1 The 'customer' table

The customer table stores information about the customer. Information like the customer id, contact information, address, and date that the customer had the first transaction can be found in this table. The information in this table can be manipulated by the administrator who can add new customer information and generate a unique customer id for each new customer. Also, fields like the 'ftransdt' or first transaction date can be used later to calculate some benefits for the customer that may come with time.

6.1.2 The 'customerorderdetail' table

This table consists of detailed information of an order placed by the customer. When the customer places an order, there can be more than one item selected and the quantities and measure of units for each item in the order can be different. All this information is stored in this table. This table can also be accessed by the system administrator in case of any inconsistencies with the customer demand.

6.1.3 The 'customerorderheader' table

This table contains the order information for all the orders in the systems (active and completed). The information in this table is obtained directly from the customer who places the order. This table works in close association with the 'customerorderdetail' table. The number of items in a specific order and the total amount for that order can be accessed from this table. This table and the 'customerorderdetail' table are essential in Sales and Finance departments of the MES system. They can be used to develop invoices for the orders and maintain the status of all orders that are in the system.

6.1.4 The 'downtime' table

This table contains a history of the downtime for the CAMCELL. It has information that identifies the machine that has had downtime, the time the machine went down and the time the machine was up and running again. Also it logs the information of the corrective actions taken. This is very important information as it not only helps in keeping a downtime log but also helps in identifying the corrective actions taken for any specific problem that causes the downtime. Thus, if the same problem occurs in the future, there is a quick way to know how to fix it.

6.1.5 The 'ftool' and the 'rtool' table

These two tables store information about the different tools required. They have information like the tool code, tool material, etc. The 'ftool' holds information for the finishing tool and 'rtool' holds information relating to the roughing tool.

6.1.6 The 'jobstatus' table

The 'jobstatus' table holds the real-time information obtained from each element the CAMCELL FMS. It has information that can be used to track the current status of an order to even a part-to-part level. It has details about the process, the order, the current count of parts within that order, etc.

6.1.7 The 'login' table

This is a table that is used to store information like the username and password of each of the end-users of the MES application. It also stores the information regarding the level of authority that is assigned to each user.

6.1.8 The 'machine' table

This table can be accessed by the system administrator. The administrator can add information about any machine that is purchased and view information for the machines that are already present in the system. It stores information like the vendor code, TPM time required and the date that the purchase was made.

6.1.9 The 'measure' table

This table contains information about the unit of measure like DZ for dozen, EA for each, BX for box, etc. This is again used as a crucial input while placing an order, printing the invoice, etc.

6.1.10 The 'pallet' table

This table contains information for the pallets used in the CAMCELL. It stores information regarding the pallet, its calibration date, the fixture that can be used with the pallet, pallet number, etc. This information can be used during the checks for the set-ups for any product that needs to be processed through the system. It gives information such as the correct fixture that is used for each pallet as a way to prevent the associate from setting up the process incorrectly.

6.1.11 The 'parts' table

This table contains information relating to each part that can be processed in the CAMCELL. It can store details about the lead-times, the process code, the sequence of operations, etc. that are necessary in order to get the CAMCELL setup correct when a specific order is being processed. It can also be used in order to print out work instructions for the associates on the floor.

6.1.12 The 'process' table

This table holds all the information necessary for each process in the CAMCELL. It stores information like the part number, machine number, station number, setup time, cycle time, etc. This again aids the associates on the floor with their job and keeps details on how the part is processed at each stage as it goes through the system.

6.1.13 The 'processlog' table

This table is very useful in determining the downtime. It keeps information relating to the process, the cycletime and date, thus making it possible to tract key metrics such as 'OEE' or "Overall Equipment Effectiveness'.

6.1.14 The 'rawmaterial' table

This table stores all the data necessary for the raw material that is used in the CAMCELL. It specifies the reorder level for all the raw material and also the vendor information and cost.

6.1.15 The 'returns' table

This table stores all the information necessary when there is any return from the customer: the date when the return was made, what problem the customer had, and how many parts were defected if any, and the corrective measures taken to handle the return.

6.1.16 The 'sequence' table

The sequence table is a very important table. This table represents information from the 'SEQ' file (see Figure x.2)

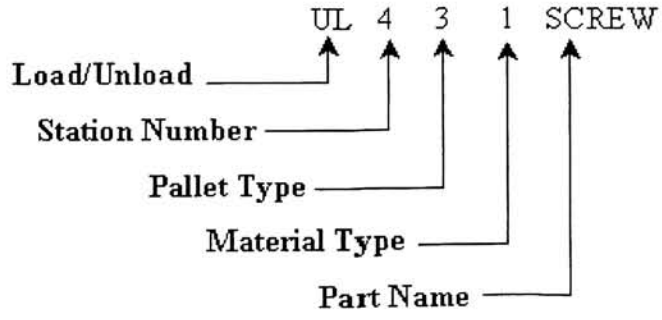

Figure 6.2 – The SEQ file

This also holds information relating each of the sequence steps with the individual machine processes. The foreign key 'prseqcd' identifies the process sequence code and identifies the individual 'SQ' files within the CAMCELLL system.

6.1.17 The 'seqload' file

This table stores information that is represented by the 'SQ1' file within the CAMCELL system. The information like partname, docking code, the station number, the 'prseqcd', etc can be accessed using this table.

6.1.18 The 'seqlathe' file

This table stores information that is represented by the 'SQ2' file within the CAMCELL system. The information like partname, lathe code, step number, line number, description, the 'prseqcd', etc can be accessed using this table. This explains the step-by-step details that a process at the lathe machine has to perform.

6.1.19 The 'seqmill' file

This table represents the 'SQ3' file within the CAMCELL system. This explains each step that is done at the milling station.

6.1.20 The 'seqvision' file

This table represents the 'SQ4' file within the CAMCELL system. This explains each step that is done at the vision station.

6.1.21 The 'seqassem' file

This table is a generic table for all the files that lie under the category of 'SQ5' to 'SQ8'. Stations 5 to 8 are the assembly stations and their respective sequence files are represented by 'SQ5' to 'SQ8' respectively. These give details of each motion of the robotic arm at the assembly stations.

6.1.22 The 'vendor' table

The vendor table stores information about the vendor / supplier. Information like the vendor id, contact information, address, and date that the vendor had the first transaction can be found in this table. The information in this table can be manipulated by the administrator who can add new vendor information and generate a unique vendor id for each new vendor. Also, fields like the 'ftransdt' or first transaction date can be used to later calculate some benefits for the vendor that may come with time.

6.1.23 The 'zipcode' table

The zipcode table consists of the city name, state and zip code.

Reference:

1. *"Controls Definition & MES to Controls Data Flow Possibilities"*, White Paper Number 3, MESA International February 2000.
http://www.mesa.org/education_center/content.asp?id=11&type=wp

Chapter 7: Manufacturing Execution System at the Advanced System Integration Lab (MESASI)

7.1. Application Overview

The Manufacturing Execution System developed for this research is called MESASI. This stands for "**M**anufacturing **E**xecution **S**ystem for the **A**dvances **S**ystems **I**ntegration Lab". The MESASI is the middleware for the CAMCELL set up of ASI Lab. It collects information from the CAMCELL control layer and massages the data into useful information for the ERP layer. The real time information from the CAMCELL setup and the client end or the customer side is stored in a SQL server database. For the data acquisition from the CAMCELL, the Siemens WinCC open database technology was utilized. This was the core subject matter for another research dissertation done in 2002-03 (see Figure 1).

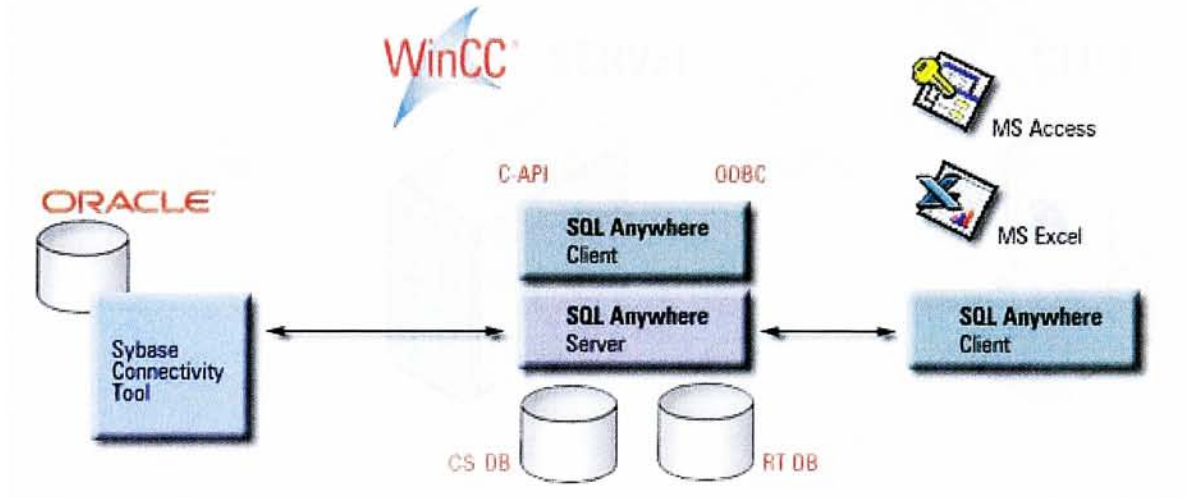

Figure 7.1 – Data Acquisition with Siemens WinCC [1]

The essence of a MES is the transformation of the available data into useful information that can be used as an efficient tool for decision making by the management. The VisualStudio.NET software by Microsoft has been used to create user interface screens for the MES built for the purpose of this research. As a part of this research, four separate modules were created. These modules represented four distinct sections within the MES. These modules within the manufacturing environment can be categorized as follows:

1. The Client / Customer Module (information that the customer is interested in)
2. The Management Module (information that the manager is interested in looking at)
3. The Operator Module (information that needs to be available for the associate on the floor)
4. The System Administrator Module (information that needs to be viewed and edited by the administrator of the system)

This application focuses on the client-server methodology (see Figure 2); where the logic flows between a client computer that is responsible for the creation of a request to get access to a particular piece of information. In response, another high-end machine which is the server acts upon the request and manipulates the information from the database to fulfill the request and transfer the required information.

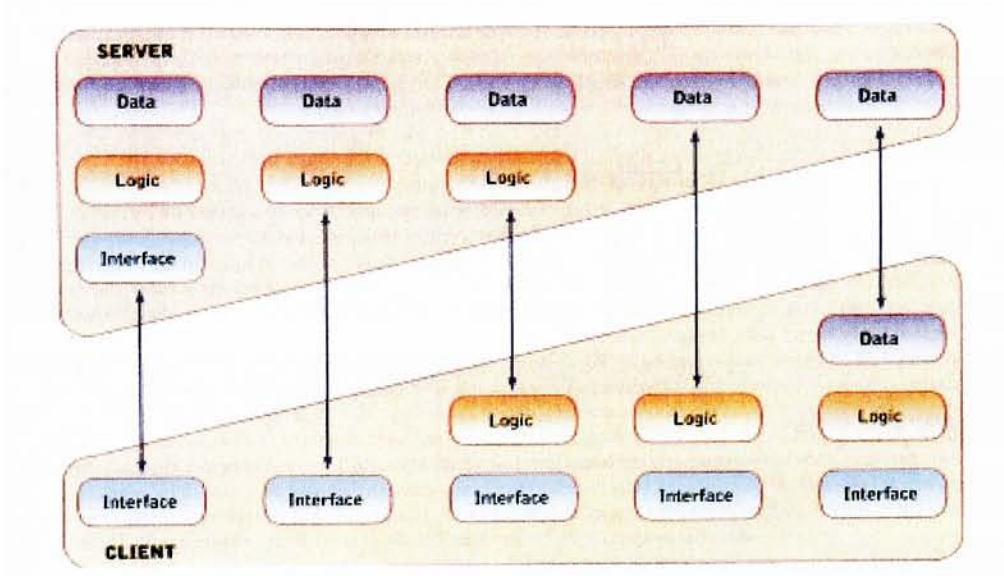

Figure 7.2 – Client-Server computing model [2]

The application starts up with the login screen (see Figure 3). Here the user of the application is prompted to login to the system using their respective username and password. Depending on the input from the user, the system validates the data and the level of access for each user, and opens the corresponding screen for further data access. Here the system accesses the 'login' table, where the three fields namely username, password and level are verified. Here the DataReader object from the .NET framework data provider is used for the verification of the data on the click-event of the submit button.

Figure 7.3 – MES application login screen

This screen has a brief explanation about MESASI and also shows two diagrams that represent the CAMCELL set up and the CAMCELL Automation Hierarchy.

7.2 The Client / Customer Module

The standard menu screen for the customer module is represented in figure 4. Out of the 12 possible tabs seen on the menu, two of the functions have been activated for the application.

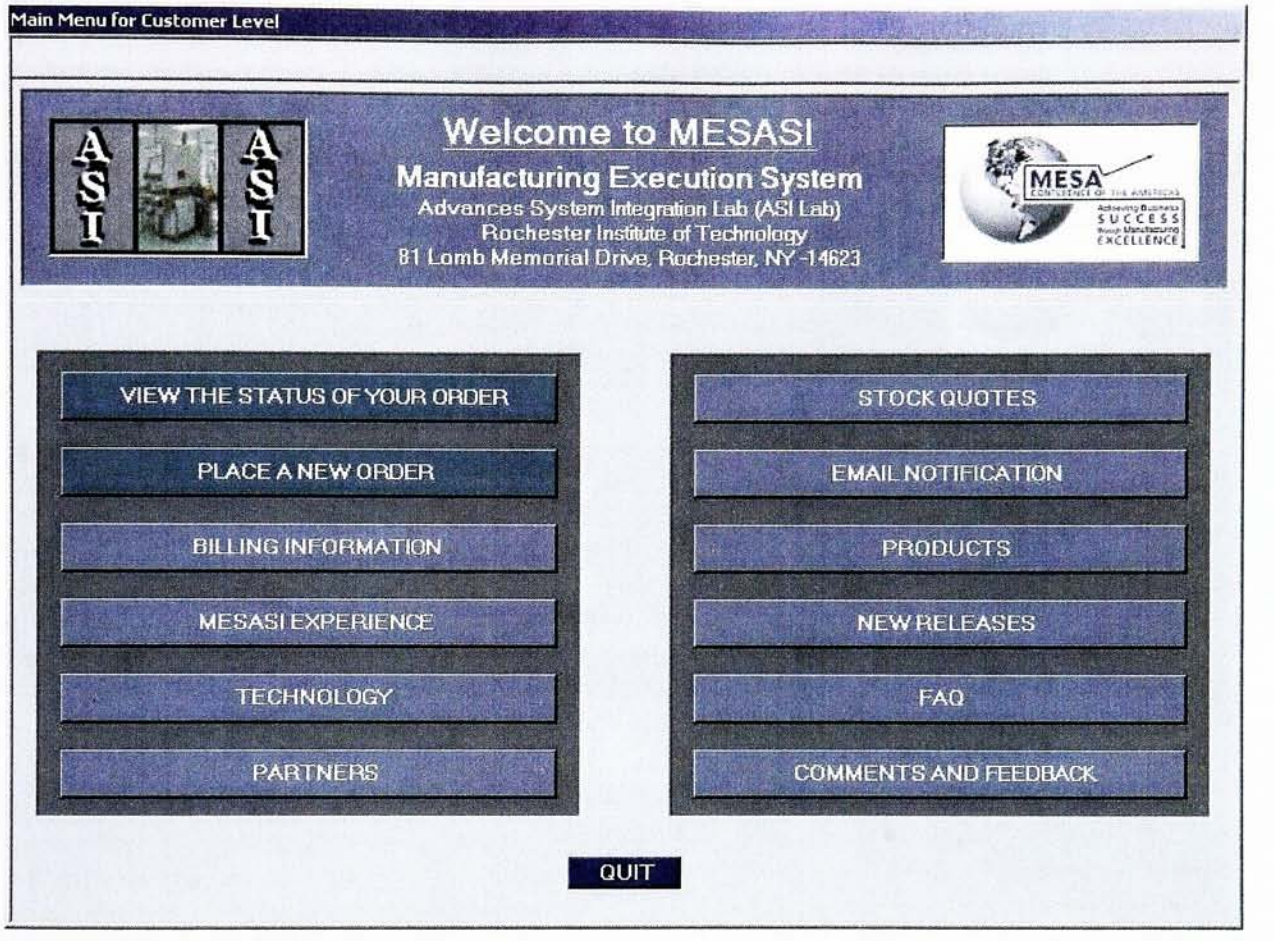

Figure 7.4 – Customer main menu

7.2.1 View Order Status

This screen (see Figure 5) allows the customer to view the status of their respective order. The customers have to type in their unique customer id, and the hit the "Click to view orders" button in order to view all their past and present orders. The application then retrieves the data from the "customerorderheader" and "customerorderdetail" table from the database that corresponds to the customer id entered. Here the DataAdapter object is used to retrieve the data and the dataset object is updated with the required information that is shown on the screen.

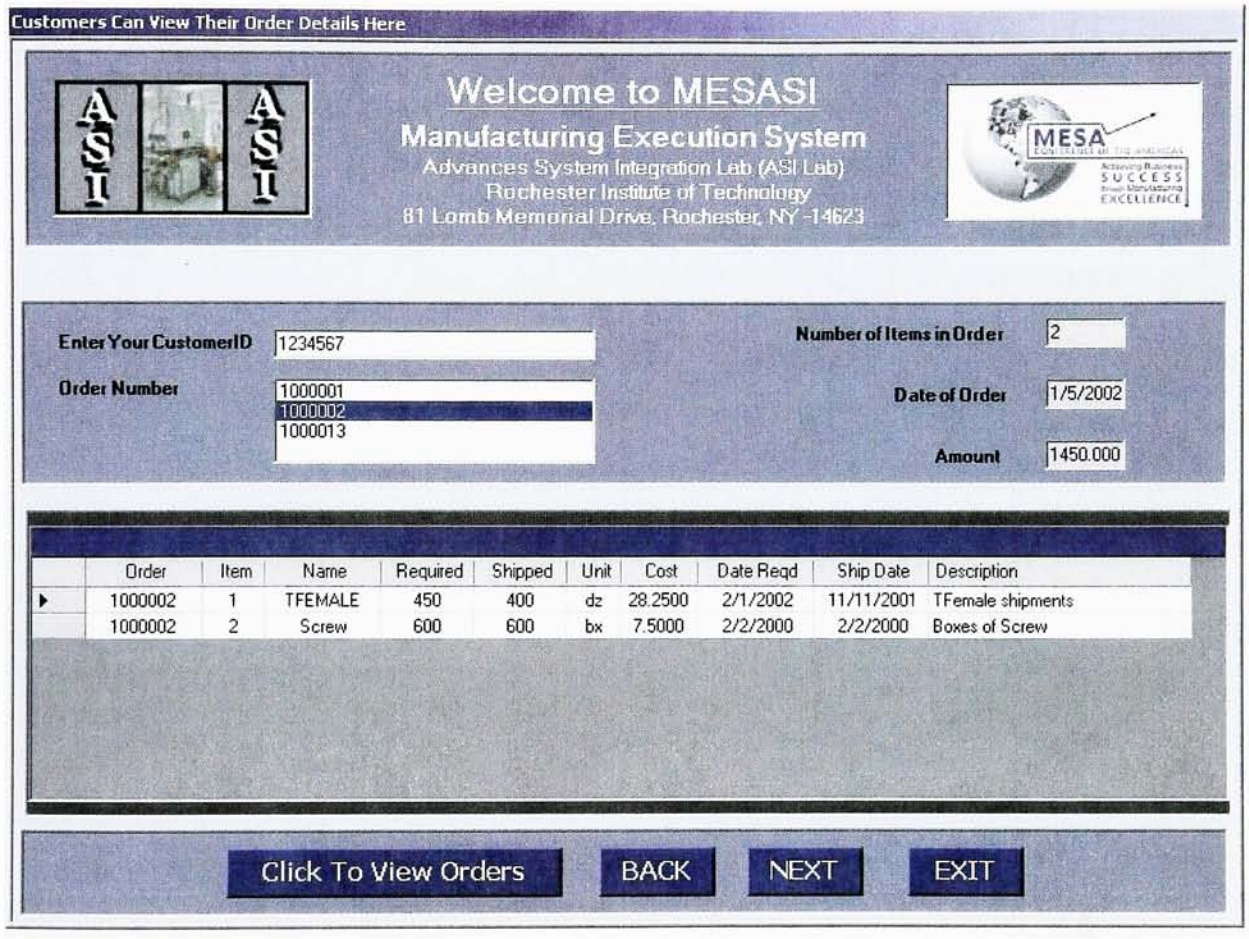

Figure 7.5 – View customer order

Here information such as 'order number', 'shipment date', 'quantity shipped', and 'unit cost' etc can be viewed.

7.2.2 Place a new order

This screen is shown in figure 7. Here the user needs to enter the customer id and then click on the "place an order" button. The application then automatically generates an order number and assigns item number '1' for the 1st item in the order. The customer then has to select the part, the unit and enter the quantity for the required item and if need be can enter another item by choosing "add more items" button. When this happens, the application keeps the same order number but increments the item number by one. Thus there is no restriction on the number of items that can be present in a single order.

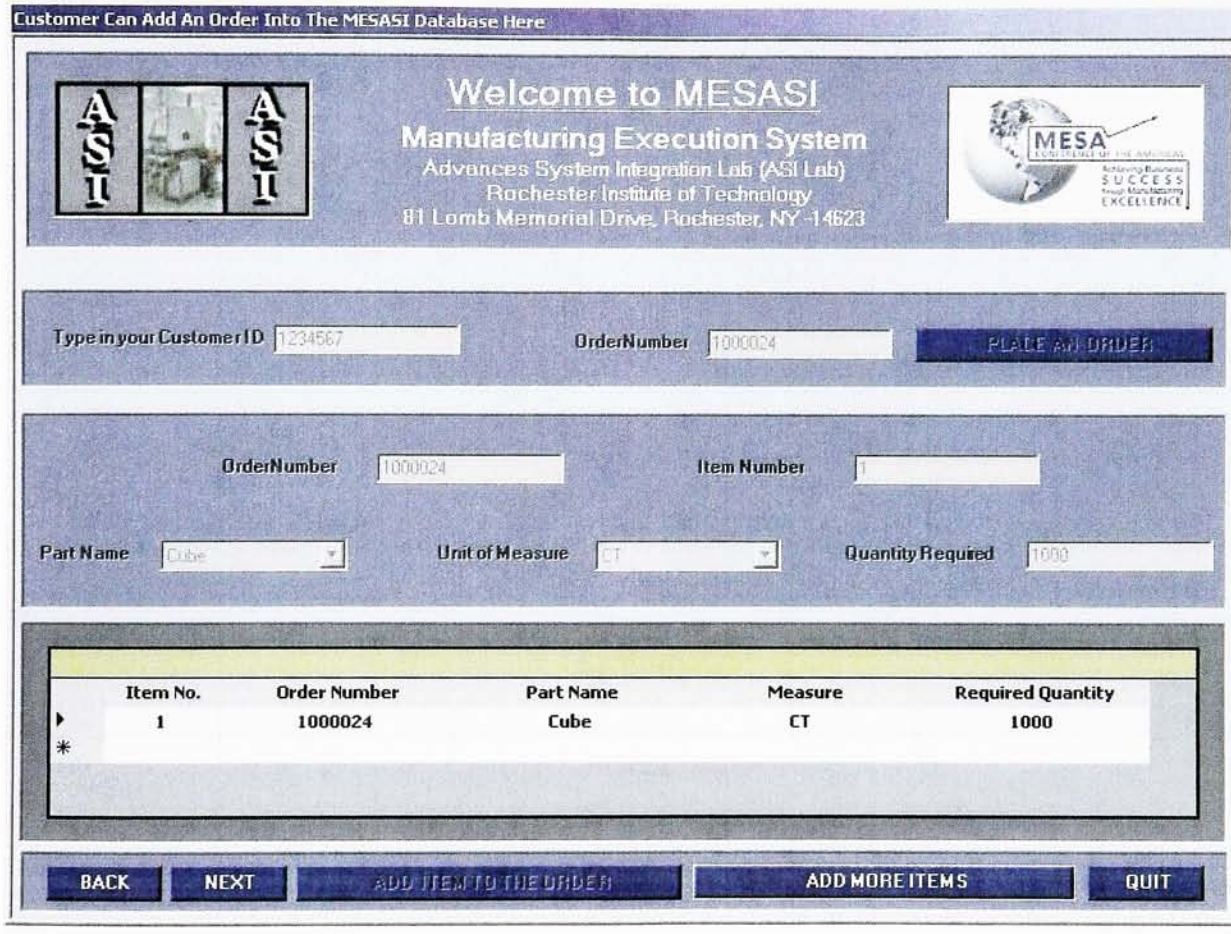

Figure 7.6 – Place an order

Here the customer can browse through all his orders using the 'back' and 'next' buttons. This screen uses the 'part' table, "measure" table, "customerorderheader" to generate information and then updates the "customerorderdetail" table with the information selected and entered by the user.

7.3 The Management Module

The standard menu screen for the management module is represented in figure 7. Out of the 12 possible tabs seen on the menu, two of the functions have been activated for the application.

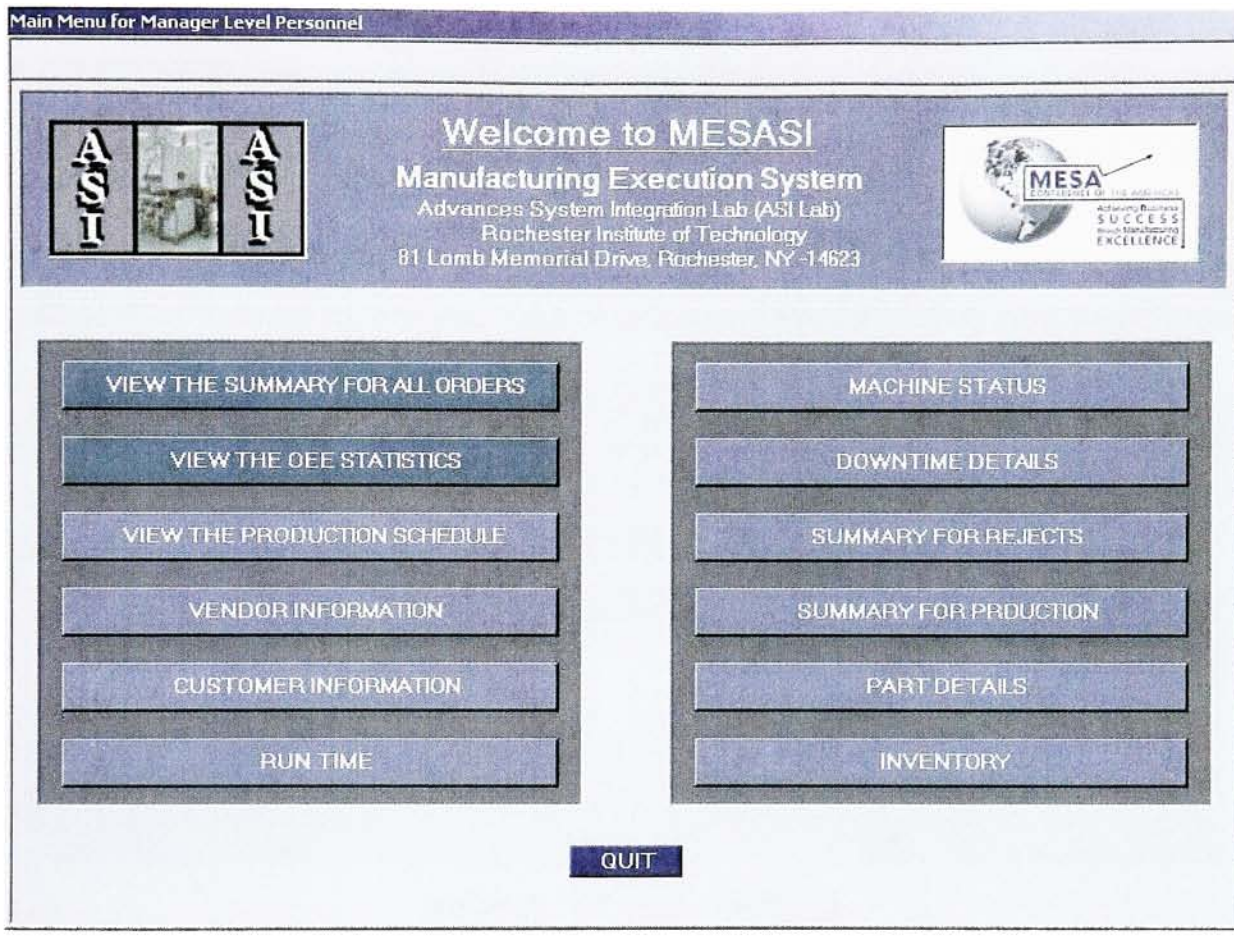

Figure 7.7 – Manager main menu

7.3.1 Summary of Orders

With this interface screen, the manager or user who has management privileges can view the status of all the orders in the system. As seen in figure 8, the manager can select the various client companies from a pull down menu. This pull down menu is populated using the DataSet and DataAdapter objects using the "customer" table.

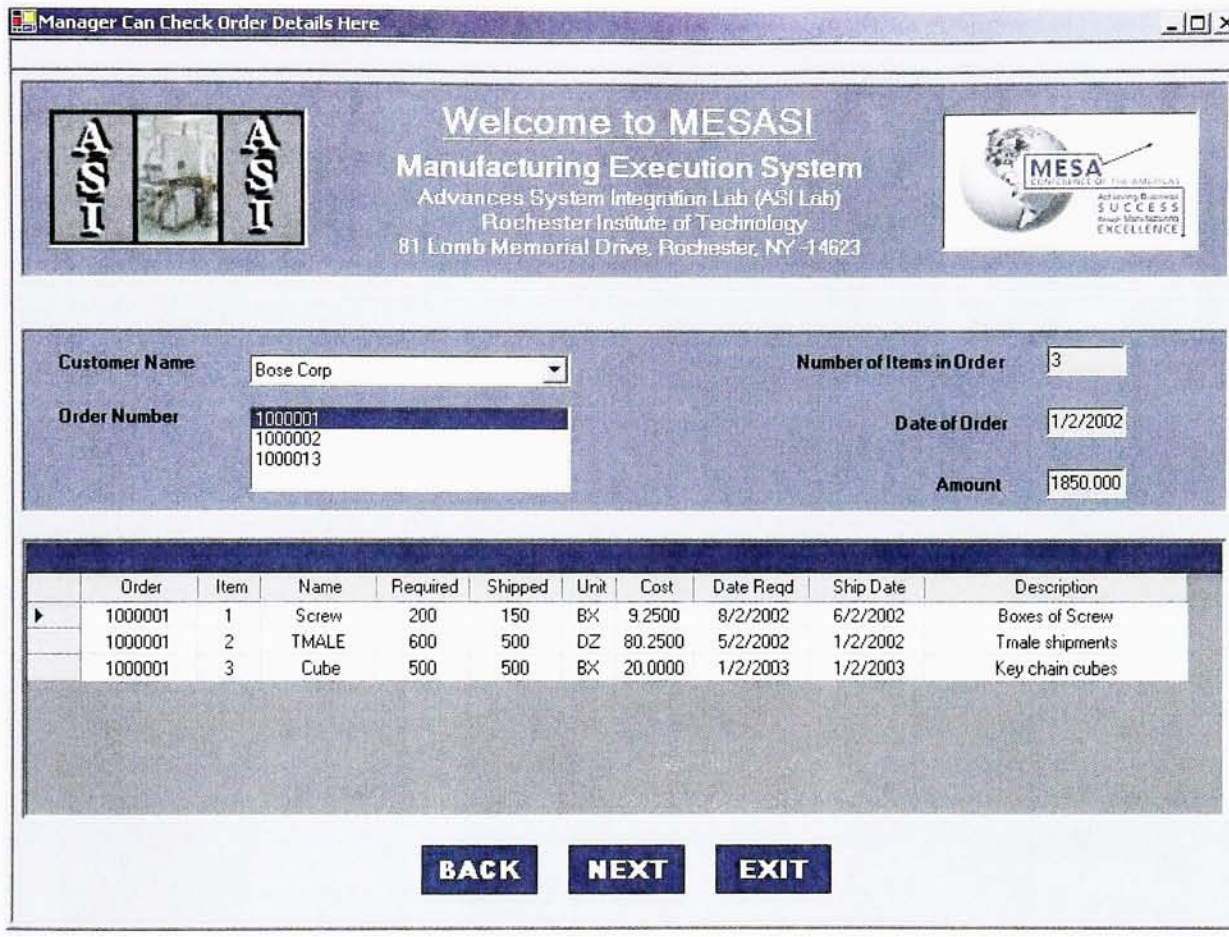

Figure 7.8 – Summary of orders

The event of selecting the company triggers the event that populates the list box with the order numbers that correspond for that particular customer. When any order number is clicked on in the list box the rest of the fields on the form and the grid get populated with the appropriate data. The order description, shipment date, cost, units, etc are shown in the data grid. The 'next' and 'back' buttons can be used to browse through the orders that show up for the chosen customer. This data is obtained from the "customerorderdetail" table.

7.3.2 Overall Equipment Effectiveness (OEE) Statistics

This interface screen presents details of the OEE measurement metric. The OEE metric is calculated for processes that run in a working day. The first step is to select the date that the user wants to see the statistics for. This populates the process drop down menu with the processes that were recorded for that particular day. This is a very effective management tool which is used to track the efficiency of the process and the equipment and can be used as a benchmark for process control. (See Figure 9)

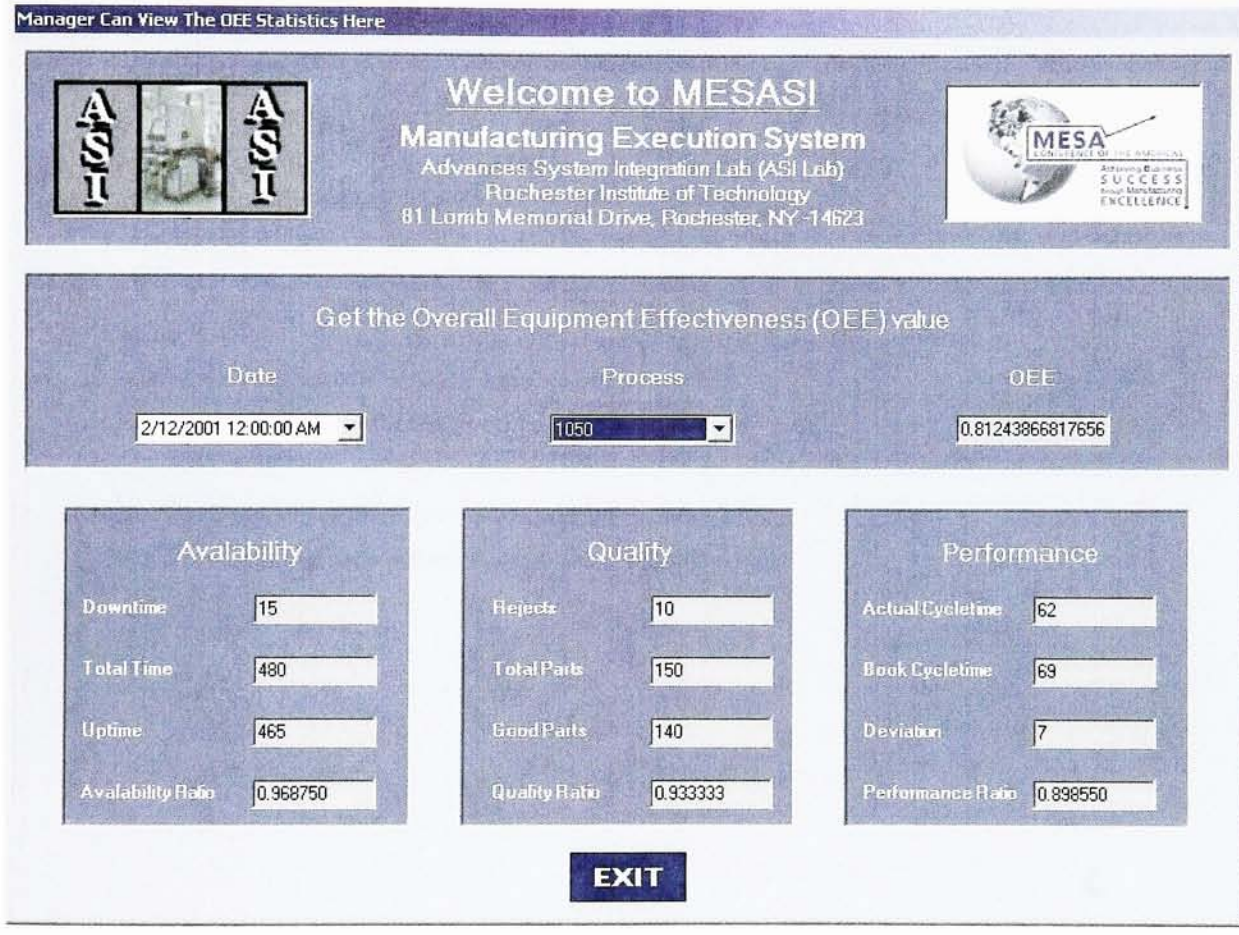

Figure 7.9 – OEE statistics

7.3.2.1 Calculations of OEE:

"OEE is simple and practical. It takes the most common and important sources of manufacturing productivity loss, places them into three primary categories and distills them into metrics that provide an excellent gauge for measuring where you are - and how you can improve!" [3].

The three categories that are defined for calculating OEE are:

1. Availability Ratio

   Availability takes into account Down Time Loss, and is calculated as:

   Availability = Operating Time / Planned Production Time

2. Performance Ratio

   Performance takes into account **Speed Loss**, and is calculated as:

   Performance = Pieces Produced / (Ideal Rate * Operating Time)

3. Quality Ratio

   Quality takes into account **Quality Loss**, and is calculated as:

   Quality = Good Pieces / Pieces Produced

The OEE metric is a product of these three ratios, and thus is calculated as:

OEE = Availability ratio x Performance ratio x Quality ratio

7.4 The Operator Module

The standard menu screen for the Operator module is represented in figure 10. Out of the 12 possible tabs seen on the menu, two of the functions have been activated for the application.

Figure 7.10 – Operator main menu

7.4.1 Production Schedule

This interface screen is shown in figure 11. This is an interface screen designed to be viewed by the operator or the supervisor on the shop floor. The application is designed in a manner that the schedule for production for any day is automatically generated by the system at the server level such that the order that needs to be shipped the earliest takes precedence. Thus, the supervisor can access the production schedule by viewing this screen and can also gain access to the various details for the sequence of operations involved to produce a particular part.

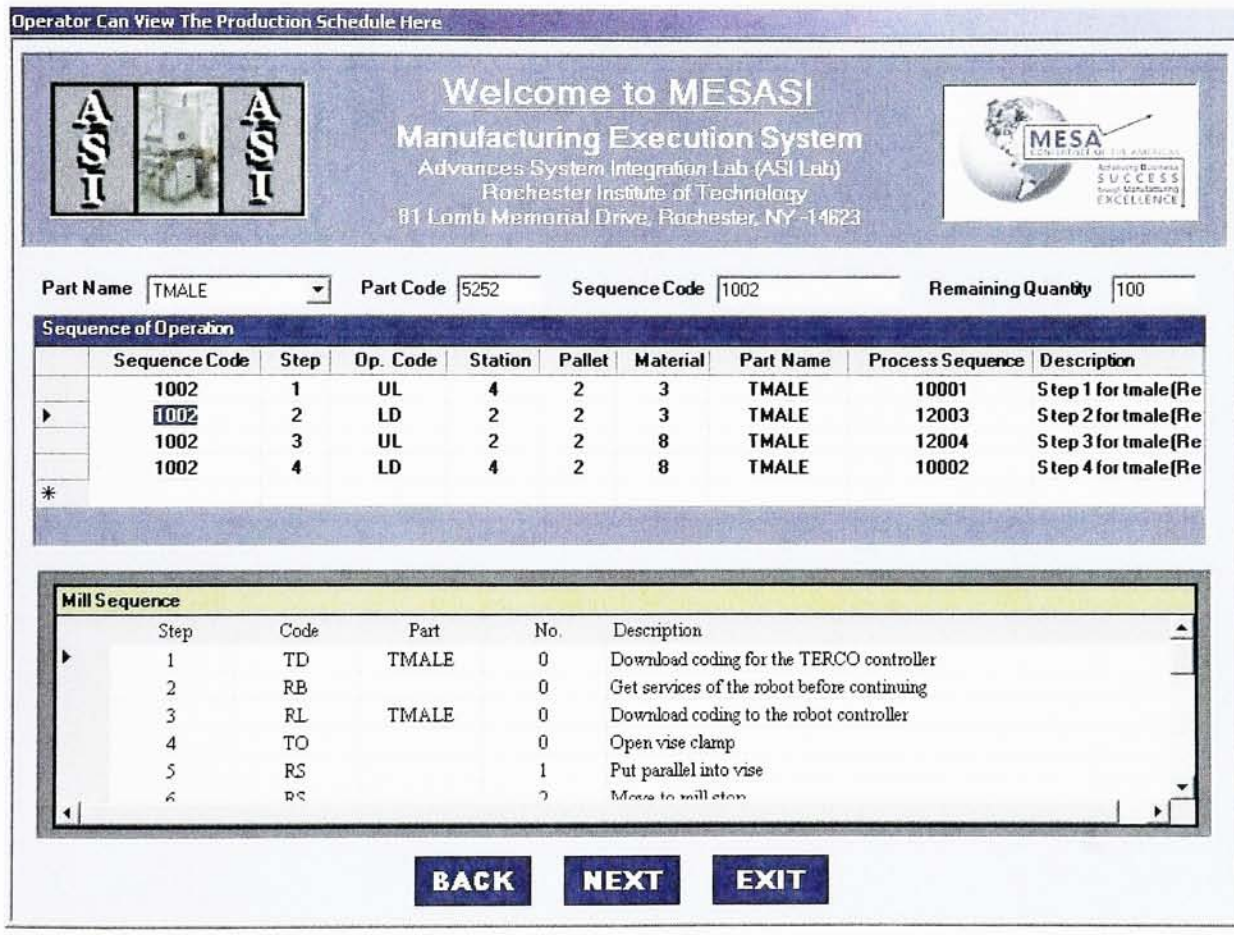

Figure 7.11 – Production schedule

This screen shows the part name, part number and the quantity that is left to be built. When the user clicks on the sequence code, the first grid is populated with the sequence file (the SEQ file). The 'back' and 'next' buttons can be used to browse through the sequence, or it can be directly accessed by clicking on the grid. This triggers an event which retrieves the data from the respective 'SQ' files, and populates the second data grid. The second data grid shows the actual sequence that needs to be followed for each individual station within the CAMCELL.

7.4.2 Process Information

This interface screen is identical to the OEE screen (see Figure 9) that shows the OEE statistics. This is used by the supervisor to keep the shop floor within the required OEE levels, and if there is a deviation from the required OEE level, he can call an alert and get the process back on the right track. This can also be used as a motivational tool for the shop floor where **H**igh **I**mpact **T**eams (HIT) can thrive off their excellent OEE levels, and can drive the rest of the plant to achieve similar high efficiency goals.

7.5 The System Administrator Module

The standard menu screen for the System Administrator module is represented in figure 12. Out of the 12 possible tabs seen on the menu, two of the functions have been activated for the application.

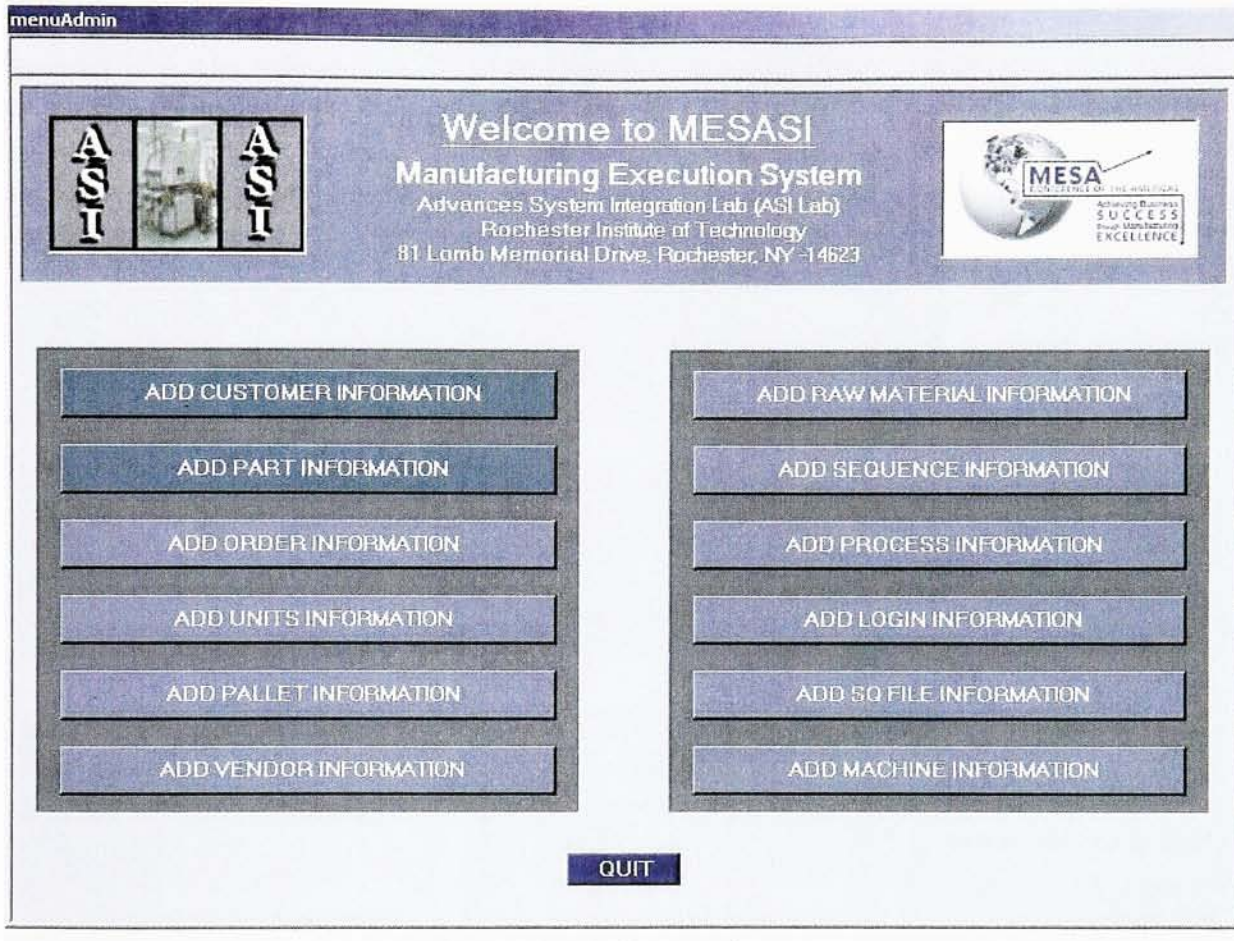

Figure 7.12 – Administrator main menu

7.5.1 Customer Information

Using the 'Add customer information' screen (see Figure 13), the system administrator can add the customer information. The administrator can also browse through all the customers that are in the database. When the user (in this case, the administrator) hits the 'add button', the module automatically generates the customer code for the database. This code is essentially the succeeding number to the largest number for the customer code from the 'customer' table in the database. Only after all the fields are completed does the save button appear active and then the user can save the new record into the database.

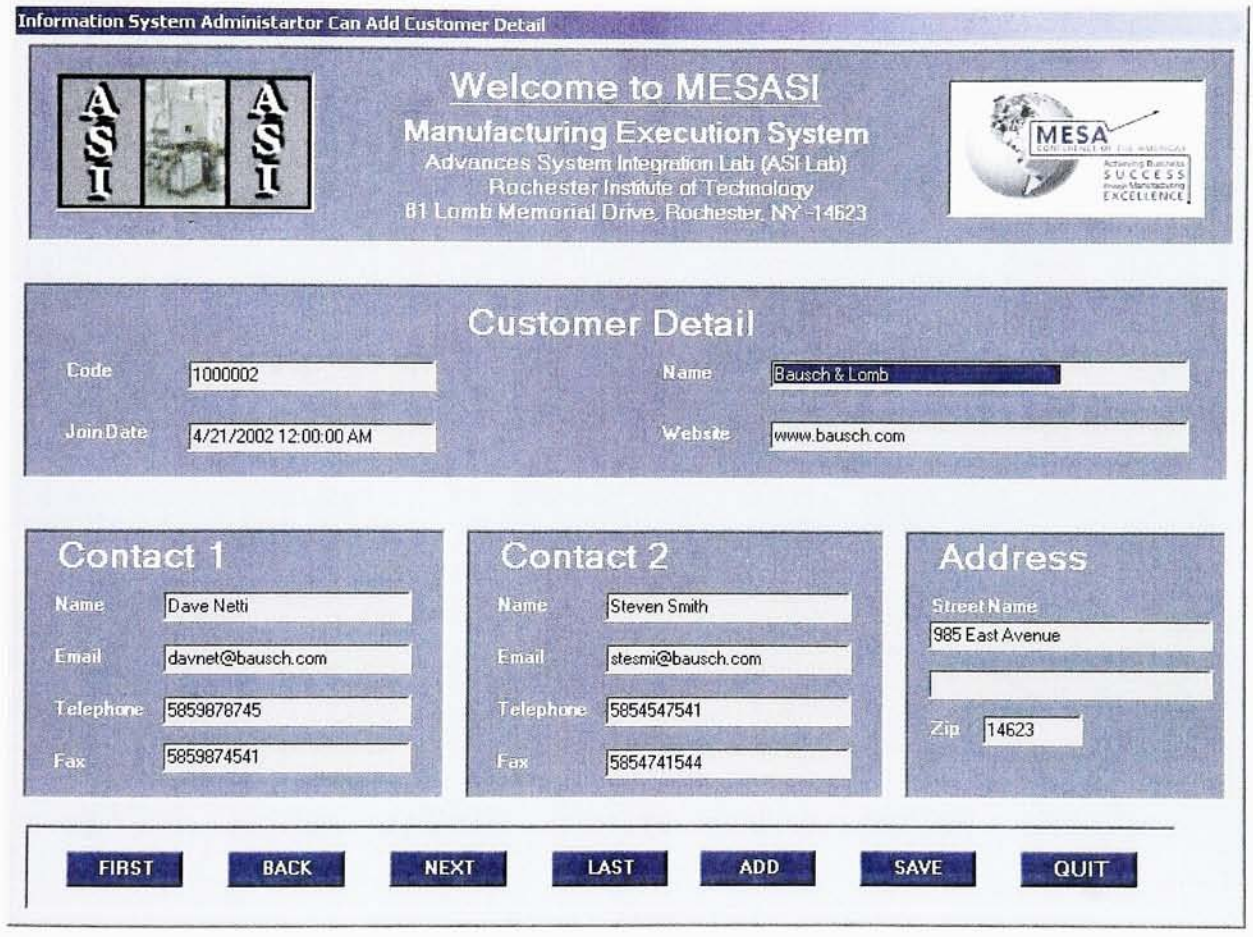

Figure 7.13 – Customer information

7.5.2 Part Information

Using the 'Add part information' screen (see Figure 14), the system administrator can add the part information. The administrator can also browse through all the parts that are in the database. When the user hits the 'add button', the module automatically generates the part code for the database. This code is essentially the succeeding number to the largest number for the part code from the 'parts' table in the database. Only after all the fields are completed, does the save button appear active and then the user can save the new record into the database.

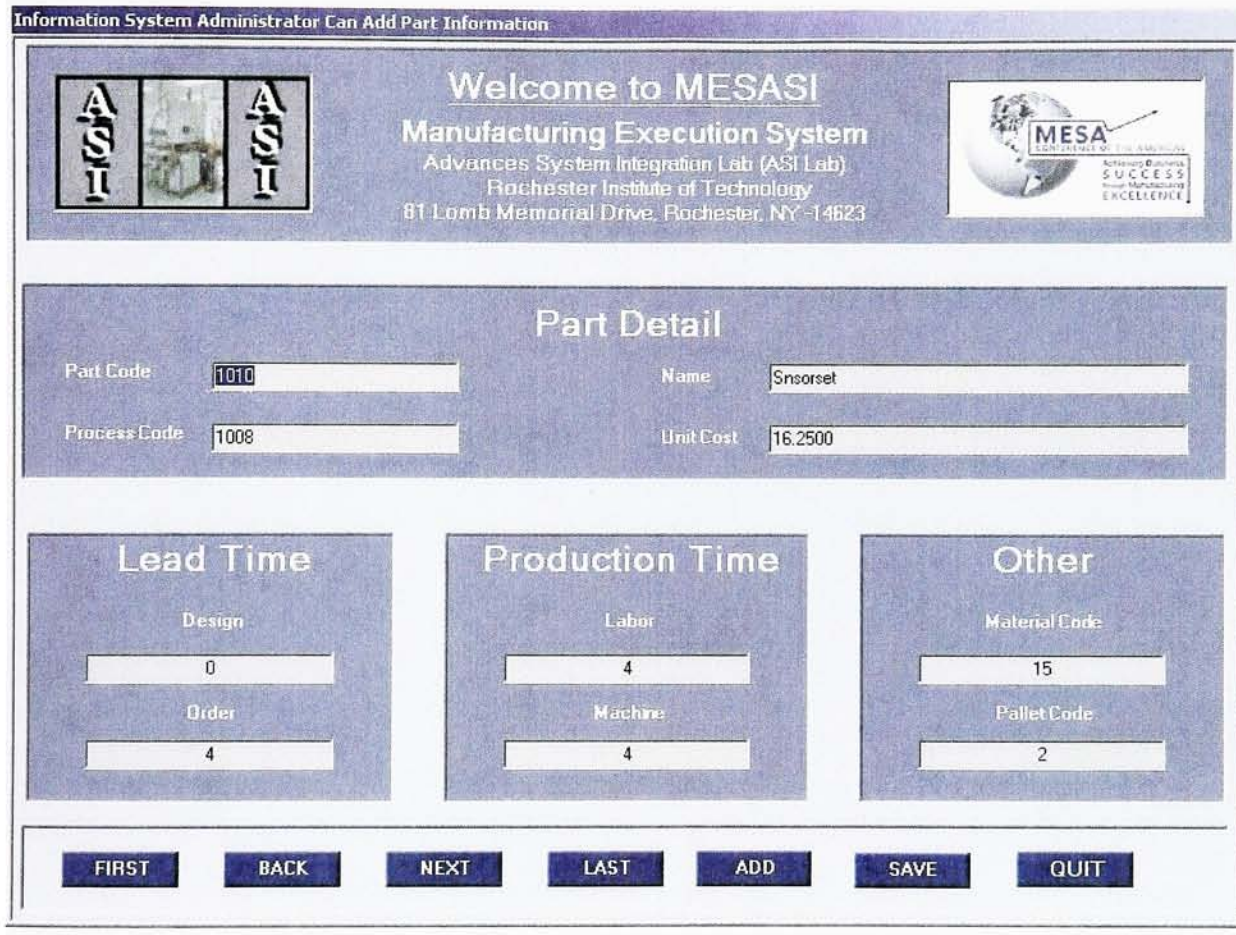

Figure 7.14 – Part information

Reference:

1. Prakash Gandhi, *"A Manufacturing Execution System using Siemens' PC Based Automation Technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2003.
2. Dr. S. R. Paidy, Course Number 0303-765-01, "Database and Information Systems", Industrial & Systems Engineering Department, Rochester Institute of Technology, Winter 2004.
3. "What is OEE?", Vorne Industries Inc., 2002-03.
   http://www.oee.com/fasttrack_oee.html

Chapter 8: Benchmarking VS.NET as an application development software

The point of reference for the benchmark was the thesis [10] that developed a MES application using Visual FoxPro. For a distributed information system application like the MES to function with optimum efficiency, certain features that are crucial are considered as the criteria for the bench mark of VS.NET as a development platform: The development platform for creating an application should certainly be very simple and easy to work with. The language in which the application is created should be very user friendly and robust. Connection with the data source should be very efficient and robust as getting timely and accurate information is crucial for a MES, the data access standard should be very efficient to make the data aggregation and data manipulation very simple, application deployment must be very adaptive to all systems and must have a capability to integrate with all possible systems through interoperability, transactions, remoting, and graphics (potential to utilize pictures and charts).

8.1. C sharp (C#)

In recent times, choices for developing applications that have the ability to deliver solutions and services in a closely-knit internet world, two main choices have arisen. On non-Windows platforms, such as UNIX and Linux, Java has emerged as the choice for developing web based solutions and services. .Net and its main languages, C# and Visual Basic, offer similar possibilities on a Windows platform. The MESASI application was developed in C#. Following are some of the advantages that C# offers [1, 2]:

8.1.1 Integration with the .NET framework

C# has a lot of features that are integrated in the .NET framework which makes C# a premier choice while developing applications in .NET. It has an inbuilt extensive forms development environment in .NET which is very useful for business application development similar to visual basic. It also brings in the concept of rapid application development (RAD) to developers of C and C++. C# also has access to thousands of classes that do not need to be recreated.

8.1.2 Safety

C# does not allow an application to run unless a variable is initialized to a valid initial value. This acts as an error-proofing technique against the shortcomings of C++ / C where the variables that are not initialized act to put the program out of sync.

8.1.3 Object oriented (OO)

C# is extremely inclined to object oriented programming. Actual strings can be treated as objects and can be used with methods such as 'ToUpper' in order to convert the string to 'ALL CAPS'.

```
1   Console.Write("hello, world".ToUpper());
```

Also, simple data types like 'int' can be converted into objects and used whenever necessary.

8.1.4 Easy to understand syntax

Header files have been removed in C#. The syntax in C# is very logical and more readable than C or C++. The event handling in C# is also very intuitive to the operation that needs to be performed aiding in better understanding. The pointers are eliminated from C#, as the compiler does not allow direct memory allocation.

8.1.5 Memory management and garbage collection

C# removes memory management issues from the developer by using the .NET garbage collection scheme. Items that are not referenced are marked for garbage collection, and the Framework can reclaim this memory as needed.

8.1.6 Using Namespaces

In C#, the developer can directly import the namespace required by using the 'USING' statement and begin using all the components within that namespace without worrying about registry lookups.

8.1.7 The flexibility to be unsafe

In C#, in case there is a situation where the developer needs to use pointers, it can still be done using the keyword 'unsafe' to mark that specific block of code. This facilitates that particular block of code to by-pass the type-safety feature of the .NET framework.

8.1.8 Inheritance

C# has the ability to inherit components and extend upon them even if they are developed in a different language. This makes C# interoperable with the other .NET framework languages.

8.1.9 XML compatibility

C# supports XML integrity within the compiler. It allows the XML documentation to be generated. This is done by using the syntax '///' followed by the XML syntax. This allows the developer to store a variety of information like hyperlinks and related document within the comment. It also provides the ability to store the XML tags and documentation as XML files that can be later viewed using the Internet Explorer browser.

8.1.10 Eliminate Buffer overflows

A buffer overflow occurs when the computer memory holds more than one instance of the same data type. Thus, an overflow occurs when data is added to the buffer outside the block of memory allocated to the buffer. In C#, this buffer overflow can be explicitly identified or ignored and can be managed internally by the compiler by not allowing space to he identified overflows.

8.2 Interoperability of .NET

As seen in the figure 8.1, Java source code is compiled into java byte code on a 'Java Virtual Machine' or JVM. In contrast to this, the C# or .NET source code does not compile to a C# byte code. It in fact compiles to a 'common intermediate language' instruction set or CIL that runs on common language runtime environment and not only explicitly on C# runtime. The CIL instructions are inside a .module file. These .module files can be combined to form .dll or .exe file. Together the .exe and .dll files are referred to as assemblies.

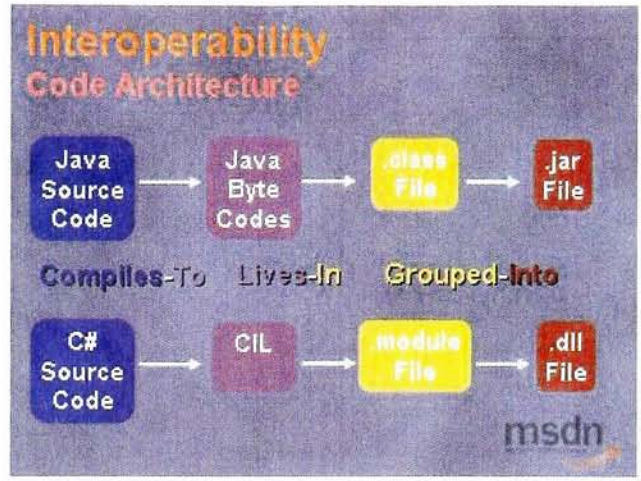

Figure 8.1 – Comparative code architecture of VS.NET vs. JAVA [3]

A common language specification is defined by the .NET framework in order to aid language interoperability. All the languages that adhere to these rules are .NET interoperable. A typical example of this is the 'keyword escape syntax rule". In every language there are certain keywords that mean something. In case a situation arises when two different languages are trying to talk to one another and that communication bridge has a keyword from one of the languages, it creates a problem. In order to work around this problem the common language specification uses '@' before the keyword and that prevents the compiler from reading that word as a keyword and thus the communication can take place. This feature is not accessible in JAVA. Thus, it can be seen that CLS and CIL together can provide true language interoperability [3].

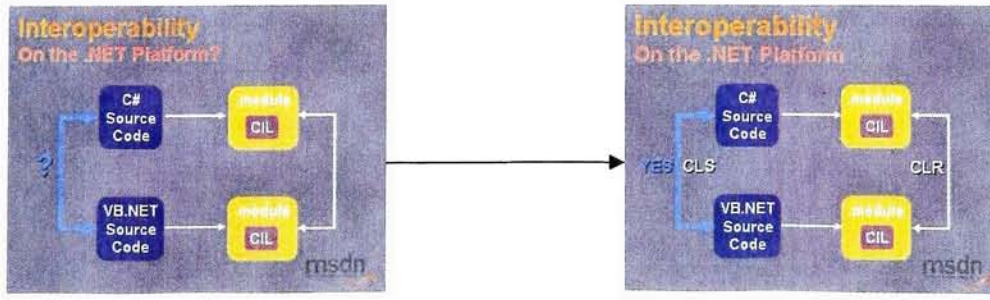

Figure 8.2 – CLS and CLR [3]

Thus, an application can be developed in several languages with different modules written in different languages, but they can all be complied in the CLR and deployed as one application of .NET as seen in figure 8.3.

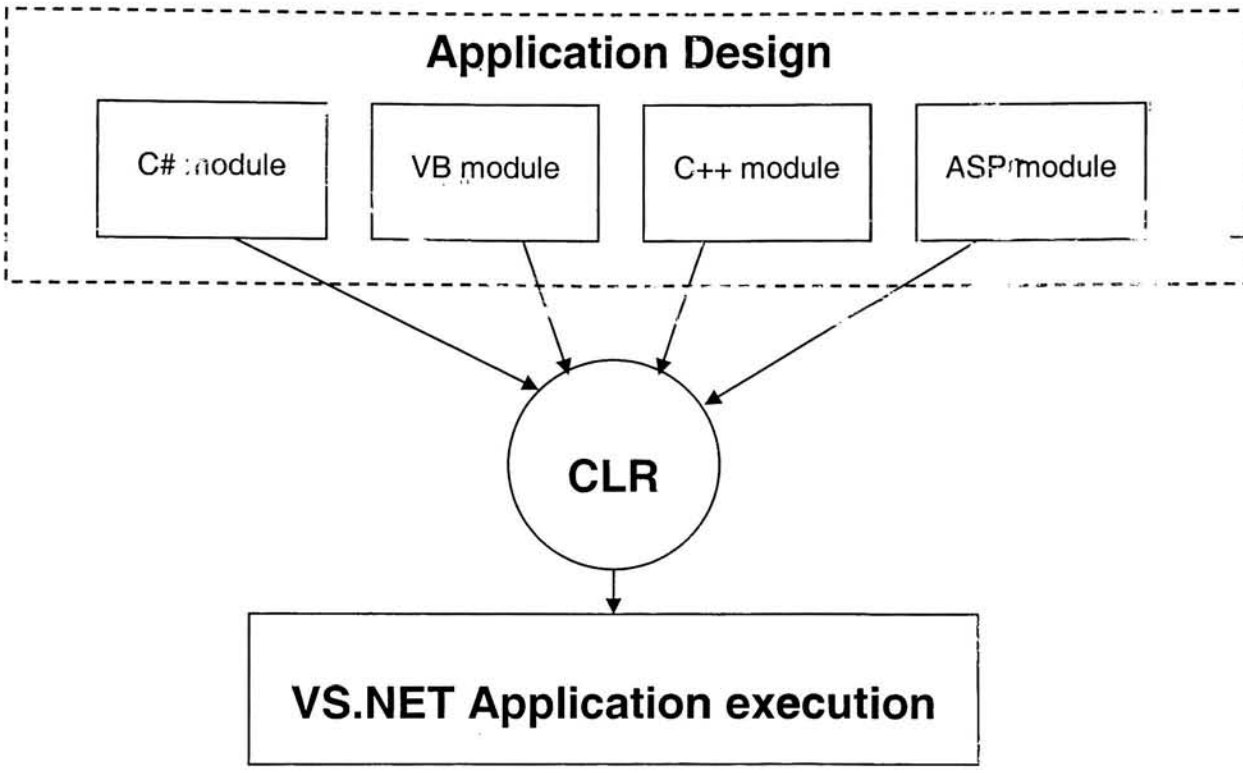

Figure 8.3 – Interoperability with .NET

8.3 Application Deployment

In the legacy windows systems, the source code is compiled into a DLL and used by one or more applications (see Figure 8.4). Now, when there is a specific need that is generated due to which the original source code needs to be changed or some new code needs to be added to the original one, the code is edited as required and then compiled again to create a new DLL. This new dll has to replace the old DLL and prompts the deletion of the old DLL. But in some cases the change may adversely affect some applications, and they might want to revert to the old DLL. In this case there needs to be a compromise at the developers end since some of his end-users would remain dissatisfied. JAVA also has similar backward compatibility issues.

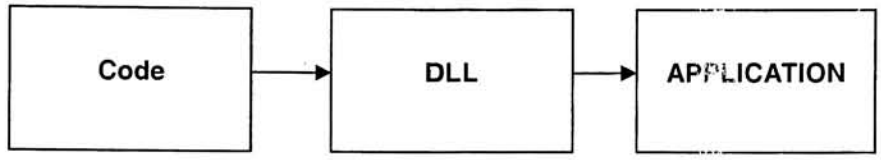

Figure 8.4 – Deployment in legacy windows systems

In the .NET framework, the code is compiled into an assembly and the assembly is used by one or more applications. This assembly has a version number associated with it. In addition to this, due to the XML integrity provided with .NET, there is an XML configuration file associated with each of the applications using the assembly (See Figure 8.5). This is where there is the information regarding the location of the assembly file and the specific version number of the assembly file.

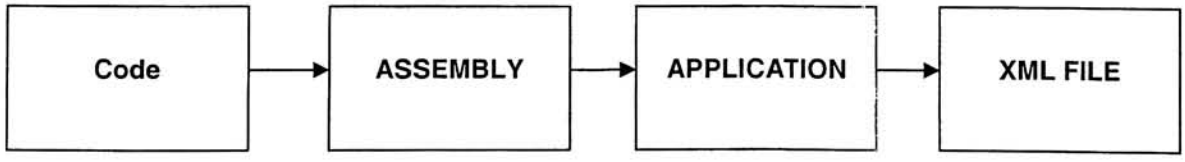

Figure 8.5 – Deployment in .NET framework

Thus, in case there is a scenario where more than one application uses the same assembly, but the upgraded version of the assembly does not work well with some of the applications. The information in the XML configuration file for those applications can be edited and the old version of the assembly can be assigned to that application such that all the applications are being executed at their maximum efficiency. Thus, this backtracking is be possible in a .NET framework since the assemblies can reside side by side in a location in the computer called as 'Global Assembly Cache', or 'GAC' (see Figure 8.6).

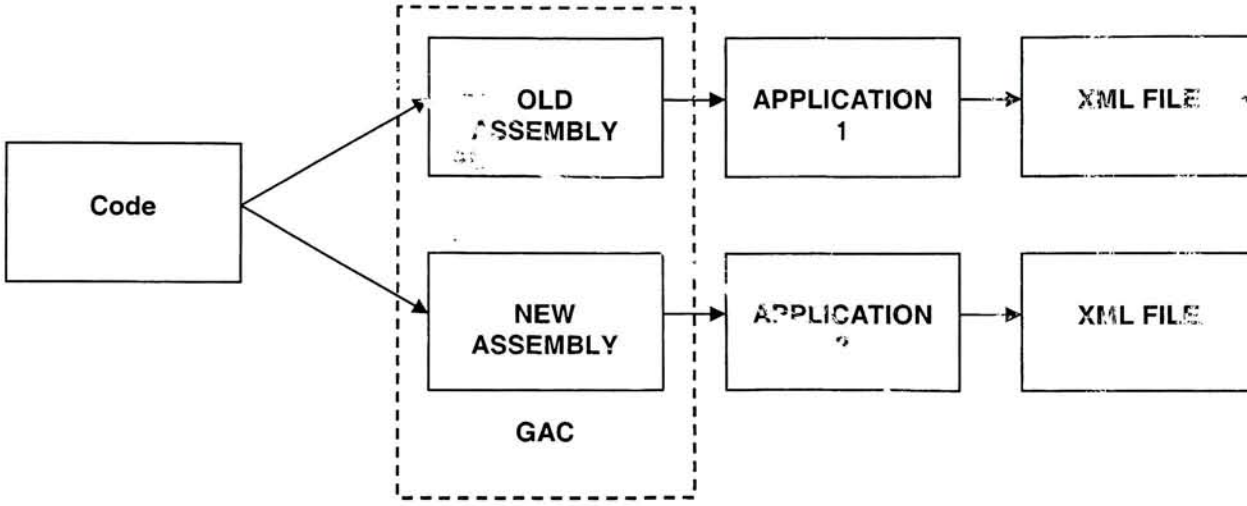

Figure 8.6 – GAC in .NET framework

8.4 Connection with a backend server

8.4.1 FoxPro Connection to the Server

In the legacy DBMS systems like FoxPro development environment, the connection to the server is via the ODBC channel. In order to establish this connection and get access to the server, a 'DSN', or 'Data Source Name', has to be created (see Figure 8.7 and 8.8).

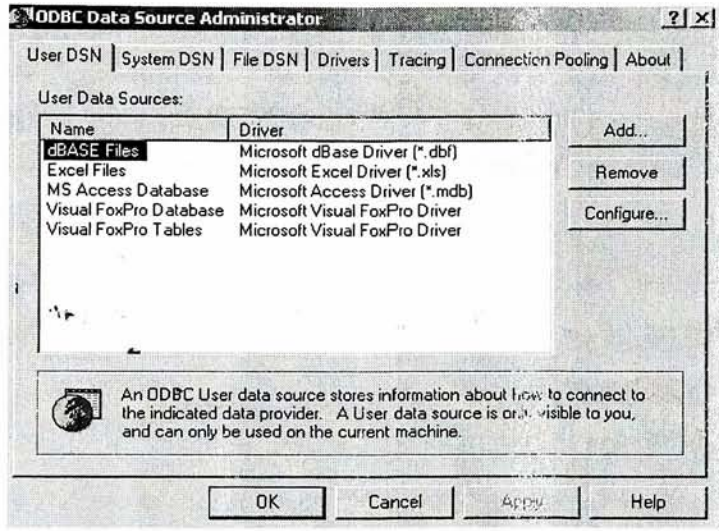

Figure 8.7 – Creating a DSN

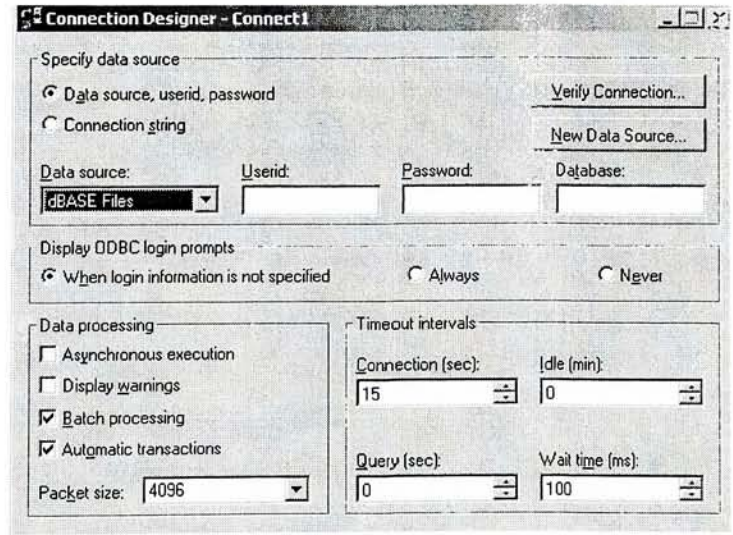

Figure 8.8 – Connection designer in Visual FoxPro

Listed below are the three different types of DSN that are used:

1. **The User DSN**
    - A User DSN is accessible only to a specific user from a specific machine on which the DSN is created.
2. **The System DSN**
    - The System DSN is accessible to all the users of the machine and also includes the NT users or network users.
3. **The File DSN**
    - The File DSN can be accessed only by the users who have the same drivers installed.

The function of the DSN is to get access to all the information that is stored in a database. It also manages the information such as the machine IP address, machine name, type of server, type of database and other information such as the security access.

8.4.2 VS.NET Connection to the Server

In contrast to this, the .NET framework uses the namespaces 'System.Data' and 'System.Data.SqlClient' in order to again access to the database. The connection is just one simple connection string. This connection string needs all the information necessary to access the server like the server name and the authentication information. In this case the data source is directly the SQL server that the developer is trying to gain access to. This can be understood using the following block of code:

- SqlConnection myConn = new SqlConnection (string);
- myConn.Open();
- myConn.Close();

Thus the connection is established using the SqlConnection object and eliminates the need for creating a DSN (see Figure 8.9 – Figure 8.10). It uses the service providers to connect directly to the database. Thus, the connection object provides a pass through for the ADO.net data access. This makes the security aspect of the application completely manageable on the remote server end, which is an advantage. The privileges and access to various tables within the database can be set on the server, and then depending on the end-user the connection object would get only that information that is allowed for that particular user.

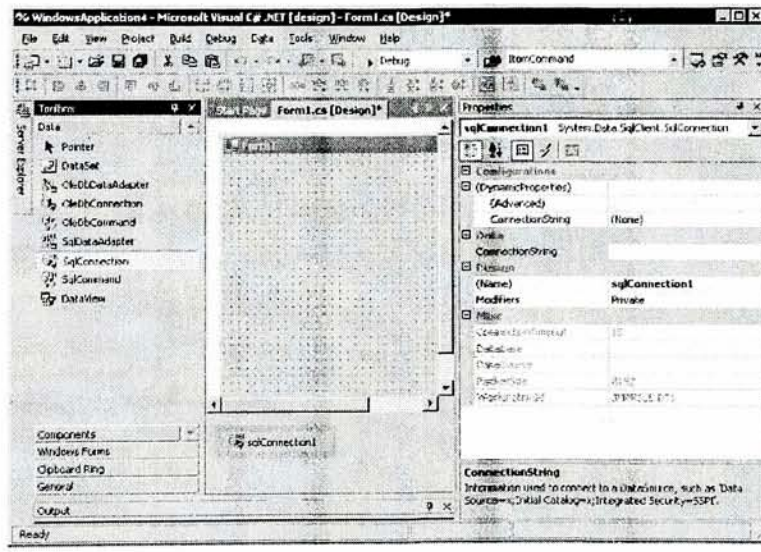

Figure 8.9 – SqlConnection object in the data tab of the toolbox in VS.NET IDE

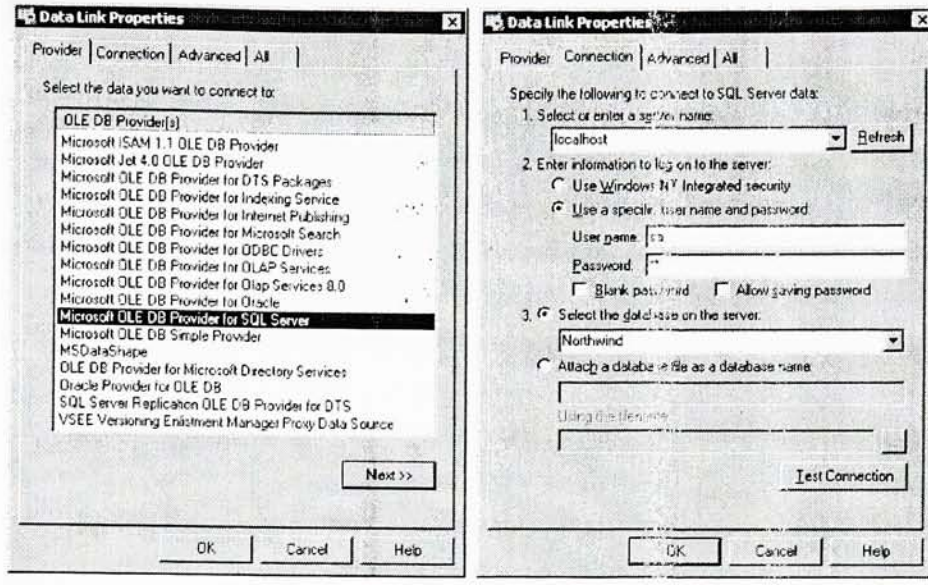

Figure 8.10 – Identify the service provider and establish the connection directly to the SQL server

8.5 Transactions

Transactions in the .NET framework can be viewed as blocks of database actions that need be performed as a single unit. Using a transaction gives the application the ability to abort (roll back) all changes executed from within the transaction if any errors occur during any part of the transaction process. For example, consider the case of the MESASI application with two tables, customerorderheader and customerorderdetail. When an order is added to customerorderdetail table with the item number and the order number, the corresponding value should be added into the customerorderheader table with the order number and the items count.

If an update to the customerorderdetail table was successful and an update to the customerorderheader table failed, the integrity of the data would be compromised. In order to guarantee that both tables would be updated successfully, the two Update data commands could be packaged into a single transaction. If one table updated successfully and the other table did not, the entire transaction can be rolled backed and all the commands would be discarded at the data source. Also, the issues that caused the failure can be corrected and the transaction can be tried again. The transactions need the 'System.Data' and 'System.Data.SqlClient' namespaces and can be implemented using the 'SqlTransaction' object [4].

8.6 GDI +

GDI stands for 'Graphic Device Interface'. GDI+ consists of the set of .NET base classes that are available to carry out custom drawing on the screen [5]. These classes arrange for the appropriate instructions to be sent to graphics device drivers to ensure that the correct output is placed on the monitor screen (or printed to a hard copy). Thus, the application does not interact with the graphic device drivers, but the GDI does. VS.NET makes the graphics programming much simpler and has a lot of support available for the GDI applications.

The services provided by GDI+ can be broadly classified into 3 categories [6]:

1. **2-D vector graphics**
   - 2-D vector graphics involve drawing of the basic features such as lines, dots and arcs that can be specified on a coordinate system. GDI+ provides classes that store information about the primitives themselves, classes that store information about how the primitives are to be drawn, and classes that actually do the drawing. For example, the Rect class stores the location and size of a rectangle.
2. **Imaging**
   - These aid in storing and rendering the pictures that are difficult to be defined using the primitives. These pictures are stored as bitmaps.
3. **Text Rendering**
   - Text rendering is concerned with the display of text in a variety of fonts, sizes, and styles.

8.7 Query builder to generate SQL queries

Generating the necessary or correct SQL statement is a task with which not everyone is very comfortable. Concepts such as inner-joins and outer-joins are not very easy to understand and develop upon. VS.NET development environment makes this task as simple as clicking a few buttons.

While configuring the SqlDataAdapter, the .NET framework has a functionality called as[omit AS] the "Query Builder" that can be used. It is very similar to the Local view and Remote view concept of a software package like FoxPro. The Query Builder screen can be seen in Figure 8.11.

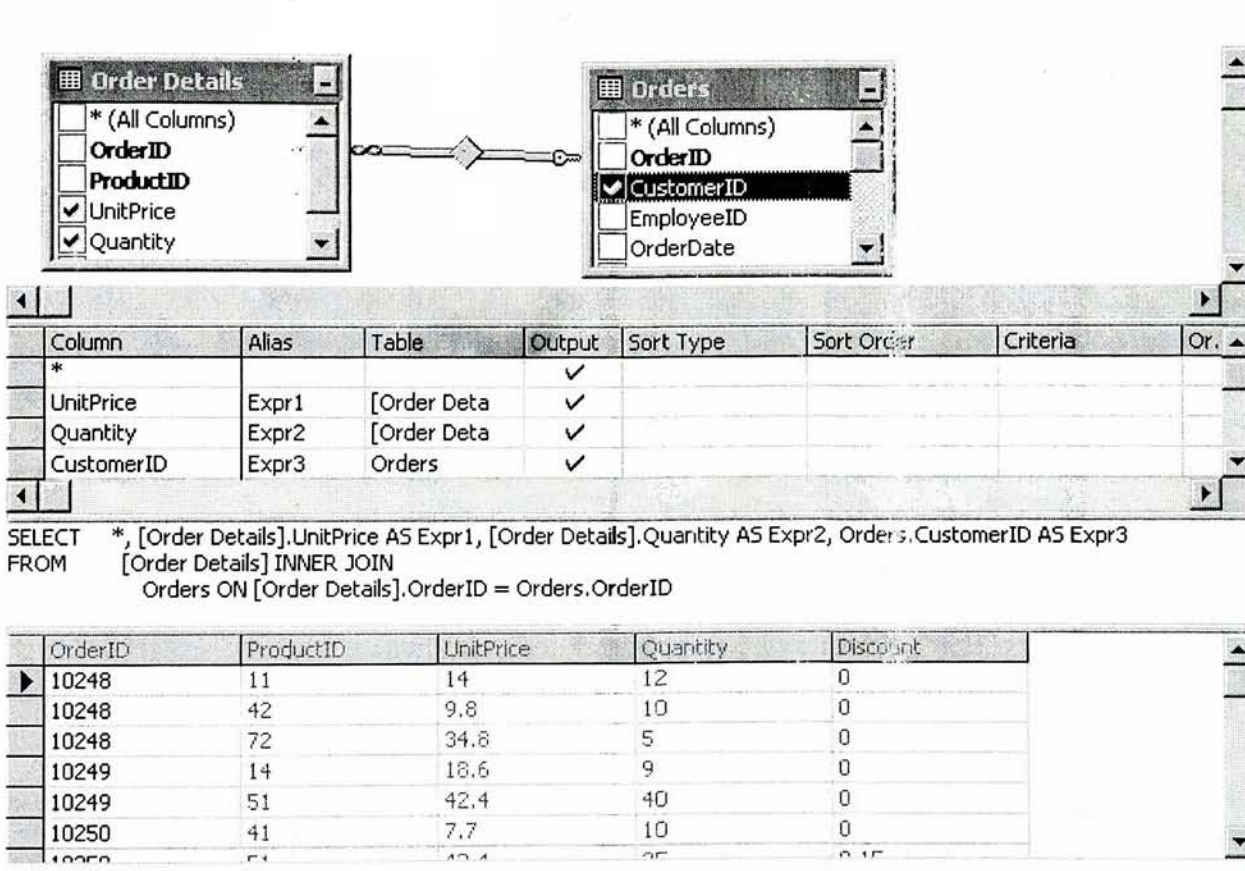

Figure 8.11 – Query Builder in VS.NET [7]

Once the connection is successful and the IDE has access to the required database, the query builder lets the developer select the tables that are needed in order to form a query. Then all the developer has to do is select the fields that are required from the necessary tables and add conditions if necessary. When the developer is done choosing the required information, VS.NET generates all the applicable SQL commands automatically. Thus, building a logical and correct query is simplified in .NET. Also, the query can be run using the 'Query Toolbar' which can be viewed from 'VIEW>>Toolbars>>Query' [7].

8.8 XML integrity with .NET Framework

XML is also called 'Extensible Markup Language', and it is a simple language that is used to represent data. The goal of XML is to provide a standard format that various applications developed on different systems can interpret and manipulate. Each XML file starts with '<?xml version="1.0"?>', and syntactically it is very similar to 'Hyper Text Markup Language' or 'HTML'. The great advantage of XML is that the developer can customize their own tags. For example '<name> John Smith </name>' defines a tag called 'name'.

In addition to the design-time compatibility, the .NET Framework class library has an entire namespace 'System.Xml' containing powerful base classes that allow reading, writing and manipulating of the XML. VS.NET has an XML designer which is very easy to work with and create and edit XML documents and XML schemas. This XML designer has three views. These views are:

1. The XML view
2. The Schema view
3. The Data view

8.9 The simplicity for application development

The .NET development environment is very adaptive for developing applications. The applications in .NET can be developed in any of the .NET languages like C++, C#, J# and VB etc. Thus, developers who are used to a particular development environment do not feel alienated when using .NET. Also, there is a variety of applications that can be developed using .NET like ASP.NET applications; Web Services, Windows Applications, Console applications, Smart device applications etc (see Figure 8.12).

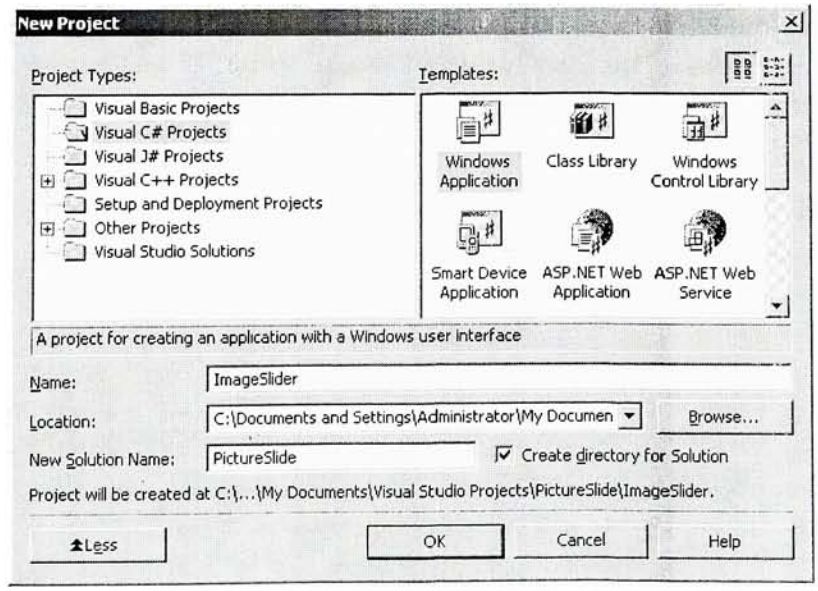

Figure 8.12 – Various applications that can be developed in .NET

8.10 .NET Remoting

.NET Remoting is a common bridge that is used by applications to communicate with one another. These applications can be on the same computer, different computers on the same network or even computers that are not on the same network. .NET remoting provides an abstract approach to inter-process communication that separates the remotable object from a specific client or server application domain and from a specific mechanism of communication. As a result, it is flexible and easily customizable. Thus, the .NET remoting supports a very efficient Distributed Applications Architecture.

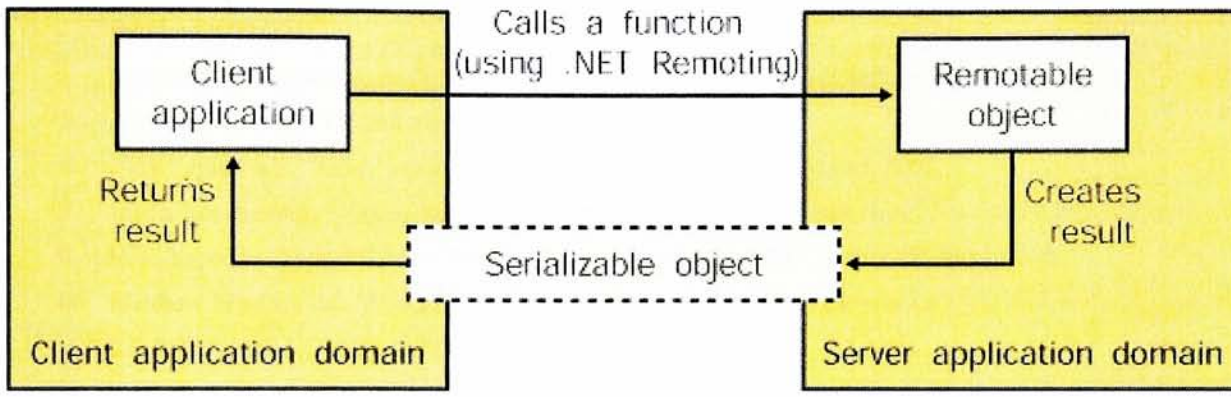

Figure 8.13 – Remoting in .NET [9]

8.11 The DataSet object in the .NET Framework

The DataSet object works with the .NET Framework as a temporary database that closely reflects the actual data in the data source. It can be looked at as the platform of communication between .NET and the 3rd party clients. The DataSet is a core part of the .NET Framework. That means that any .NET client can interpret a DataSet. The DataSet comes equipped with the capability to serialize itself to XML, which makes it easy to store a DataSet in a file, transmit it to a remote object, and even exchange the information with a client written in another language. The DataSet can hold a collection of tables and also store relationship objects between them and define column constraints (see Figure 8.14). Thus, accessing the required data becomes very easy and makes the DataSet object an ideal way for one-way transportation of the combined data.

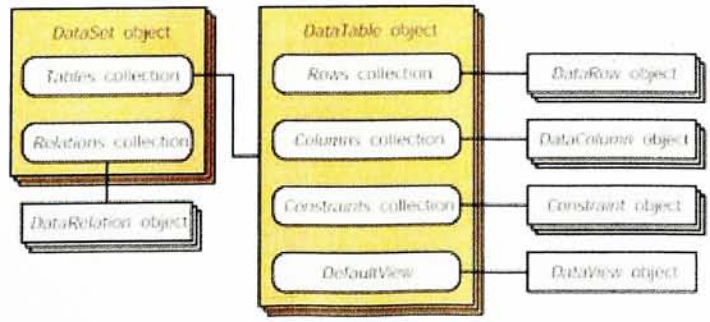

Figure 8.14 – The DataSet object in .NET [9]

Reference:

1. Craig Utley, *"Why move to C#"*, CNET Networks, Inc.4th November 2002. http://builder.com.com/5100-6390-1050356.html
2. Dare Obasanjo, *"What are the major advantages of C# over Java?"*; XTRAS.net and XDN, 22nd April 2004. http://www.kunal.org/scoble/archives/001250.html
3. Microsoft online seminar, *"Migrating from Java to C#"*, Microsoft Corporation, 18th September, 2002. http://www.microsoft.com/seminar/shared/asp/view.asp?url=/seminar/en/20020918devt1-61/manifest.xml&wmpver=8.0.0.4487
4. Michael Platt, *"Implementing Transactions with Microsoft.NET"*, MSDN library, Microsoft corporation, October 2002. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/psent.asp
5. Andrew Troelsen, *"C# and the .NET Platform"*, Apress, 2001.
6. Nick Symmonds, *"GDI+ programming with C# and VB.NET"*, Apress, 2002.
7. Julian Templeman, *"Visual Studio.NET: The .NET Framework Black Book"*, Paraglyph Press, 2002.
8. Dino Esposito, *"Applied XML Programming for Microsoft.NET"*, Microsoft Press, 2003.
9. Matthew MacDonald, *"Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting"*, Microsoft Press, 2003.
10. Prakash Gandhi, *"A Manufacturing Execution System using Siemens' PC Based Automation Technology"*, Computer Integrated manufacturing, Rochester Institute of technology, 2003.

Chapter 9: Conclusion

With the fast moving pace of technology in today's competitive industry, the choice for the software used for the development of various supporting systems is crucial. The customer today wants his products to be of the best quality possible and at the most reasonable price. The time between the order placement and order shipment has become smaller. The support systems in the manufacturing industry are responsible for providing the information necessary to handle the logistics, development, planning and control from the time when the orders are placed till the customer receives the final product. The business management layer in any manufacturing industry has to make its decisions based on the information that comes from the shop floor. Thus, the information system that supports this transfer or filtering of the information from the shop floor becomes an integral part of a successful business. A manufacturing execution system, or MES, fulfills this role of transferring the information from the control layer to the business layer in real time. Using current and aggregate data, MES guides, initiates, responds to, and reports on plant activities as they occur. The resulting rapid response to changing conditions, coupled with a focus on reducing the non-value-added activities, drives effective plant operations and processes. Thus, the software that is used to design the MES is very important. It needs to have the capability to easily integrate with the present set up and also should be able to accommodate new and improved features that the facility needs to incorporate as the situation changes.

This applied research aimed at benchmarking VS.NET as a potential candidate software platform for developing systems like the MES. It presented the importance of an effective MES in the context of the CAMCELL. The features of VS.NET were compared with those of the Visual FoxPro software, and in doing so; it presented the benefits that VS.NET has to offer.

The application that was developed (MESASI) consisted of four modules that represented different functional blocks of any manufacturing industry. These four modules were the 'information system personnel', the 'management', the 'customer' and the 'operator'. To support the interface for these modules, the data was derived from a backend SQL database, and thus the effectiveness of the client-server methodology was displayed for running the application. A SQL database was designed for storing all the data that the MES for the CAMCELL set up would require, and the database was normalized to the 3NF form. Thus, it presented the concepts of database normalization and the benefits that it provides.

For a distributed information system application like the MES to function with optimum efficiency, certain features that are crucial were considered as the criteria for the bench mark of VS.NET as a development platform.

Following are the advantages that Visual Studio.NET provides over Visual FoxPro:

The development environment for the developer is very generic and easy-to-use compared to the database centric Visual FoxPro IDE.

C# is a better option for a programming language to develop information system applications than VFP.

The need for a DSN (as required in Visual FoxPro) to develop the connection to the data-source is eliminated in VS.NET.

ADO.NET incorporates the functionality of ADO and OLE DB (used with Visual FoxPro) and has additional features like the DataSet that make it data access standard for the future.

Data manipulation and data aggregation is simplified by the 'SQL Query Builder' feature in VS.NET. Visual FoxPro lacks this kind of a step-by-step procedure to get the data.

Application deployment in VS.NET is platform independent as against a platform dependent procedure for Visual FoxPro.

CLR gives VS.NET true interoperability and thus applications can be developed in any of the languages within the .Net Framework. This is not possible in Visual FoxPro.

Unlike in Visual FoxPro, connection with smart devices is possible in VS.NET through .NET remoting.

Features such as GDI+ and Transactions make VS.NET more efficient development platform than Visual FoxPro.

VS.NET environment provides the unique opportunity to develop similar windows based application and deploy them over the internet as 'web services'.

Thus, looking into the future with improvements in technology all over the manufacturing industry, VisualStudio.NET is a very beneficial option of software for developing information system applications such as a manufacturing execution system.

Appendix A: The Picture Slide Application Tutorial

This tutorial is a step-by-step procedure to create a Windows Application called the 'Picture Slide'. This application has been created using the Microsoft Visual Studio .NET 2003 IDE. Learning through this example is a good way to get familiarized with the VS.NET IDE and the tools that it uses. A Code Project Article [1] by Rakesh Rajan has been used as a primary source for the sample code shown in this document.

This is how the application will look once it is completed and in a running condition (see Figure a.1).

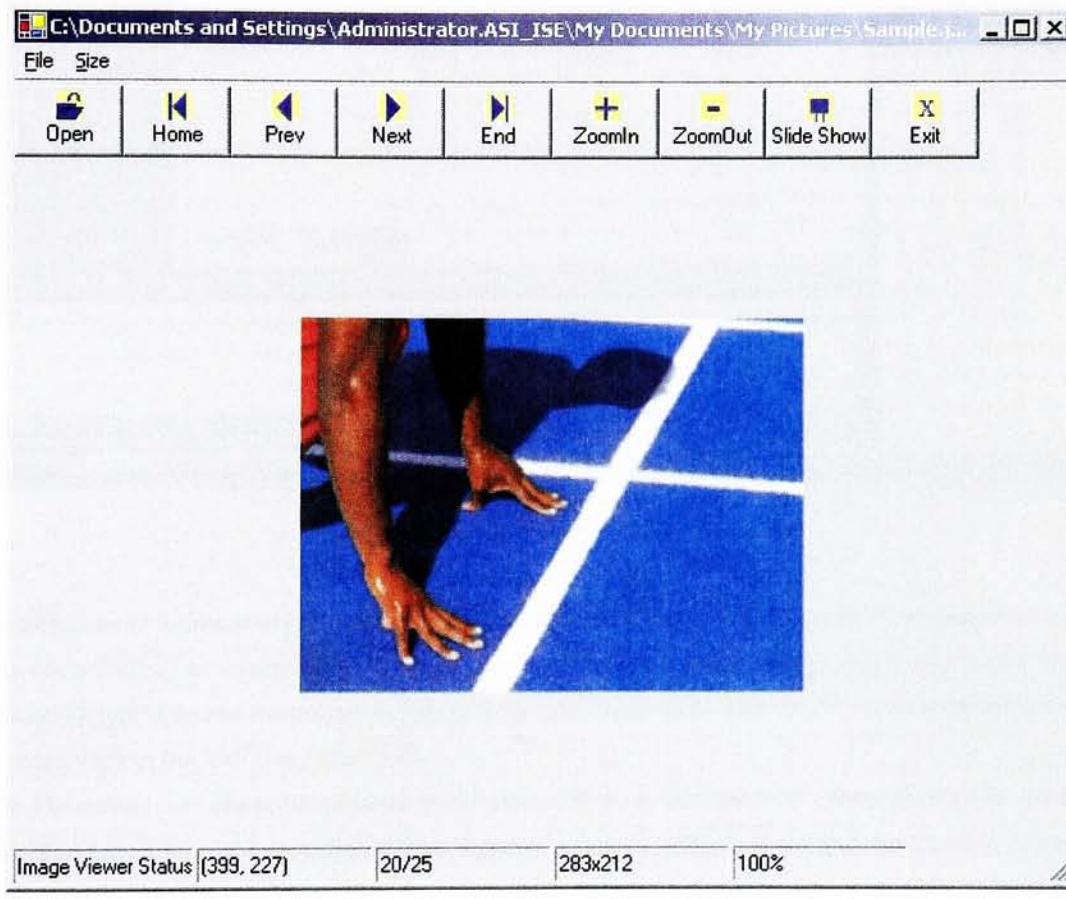

Figure a.1 – The Picture Slide Application

With this application, the user should be able to open an Image File and view it. It also provides features to 'Zoom In', 'Zoom Out', and different Sizes for the Image and also a slide show for the images within the selected folder. At any point, the user can quit the application using the exit button.

In the Visual Studio.NET IDE, start a new project by clicking on 'New Project' as seen in Figure a.2.

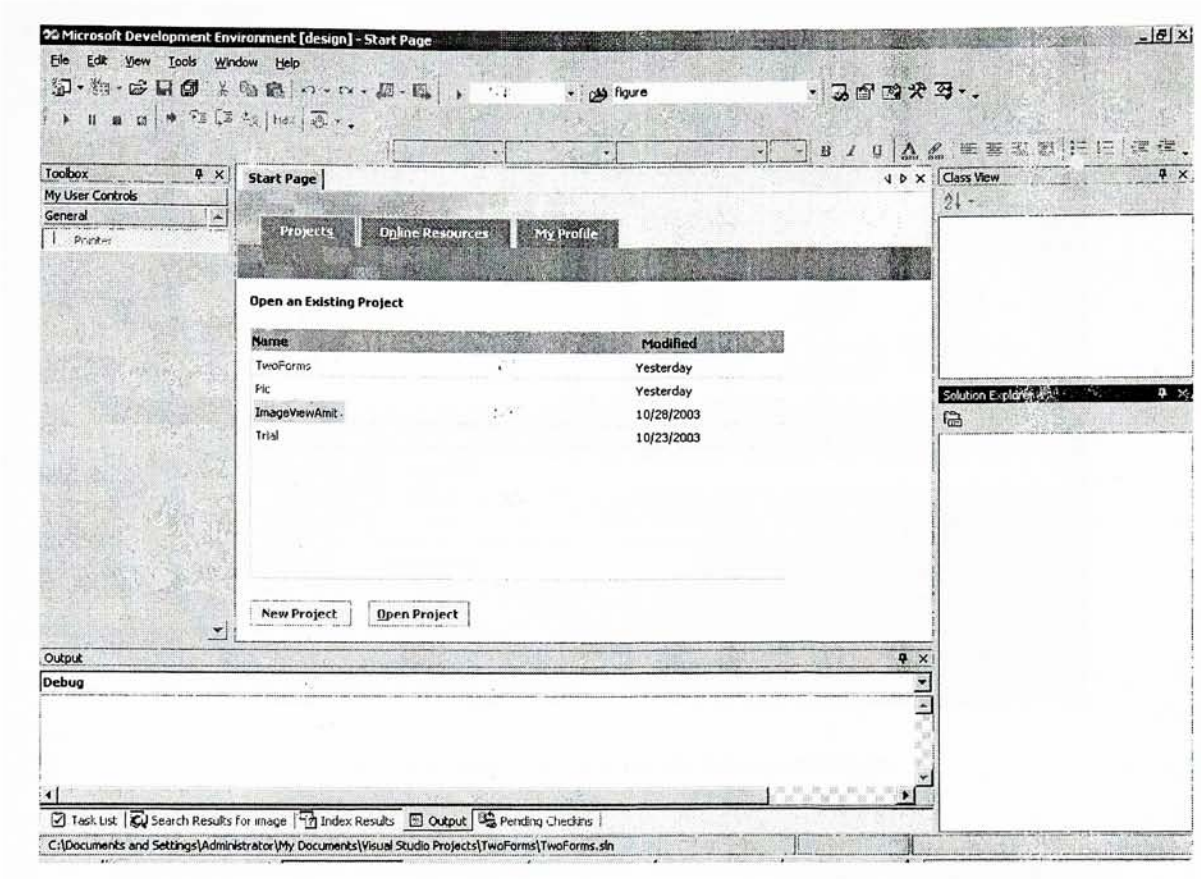

Figure a.2 – Opening a new project in VS.NET IDE

A good technique to follow is to have the name of the solution different from the name of the project that is included in it. (as one solution can incorporate more than one project). Select the Windows Application (a C# project). Call the solution PictureSlide and the project as ImageSlider (see Figure a.3). This would open up a blank windows form in the design view in the IDE (see Figure a.4).

(***NOTE**: The names that are to be assigned to the objects that the designer will create during this application are underlined and **must** be typed in as shown (case-sensitive), as this will aid in recognizing the code snippets that are to be introduced in the application.*)

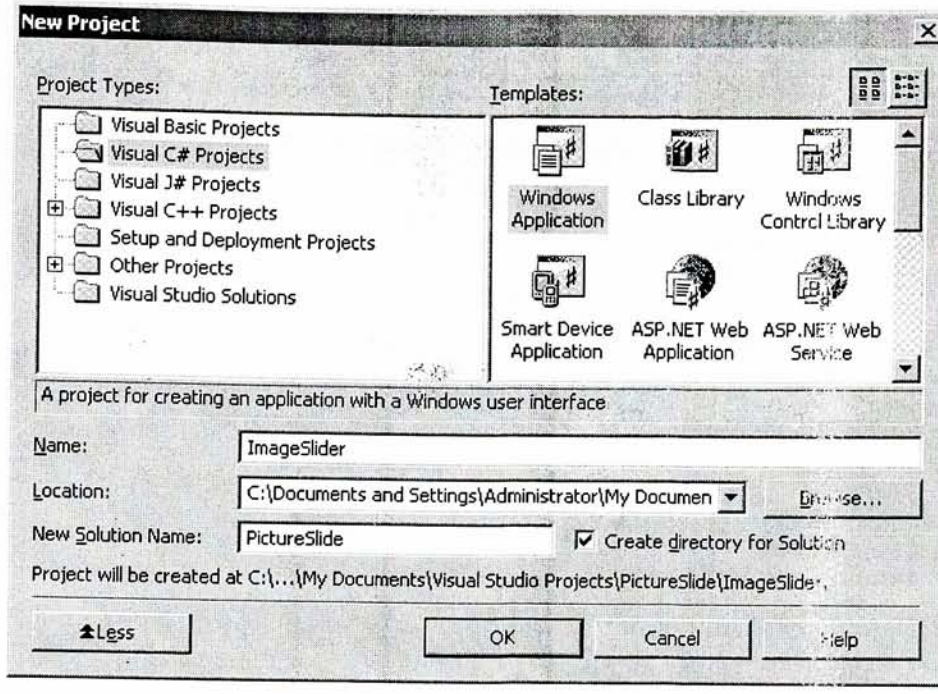

Figure a.3 – Define the Name for the Solution and Project

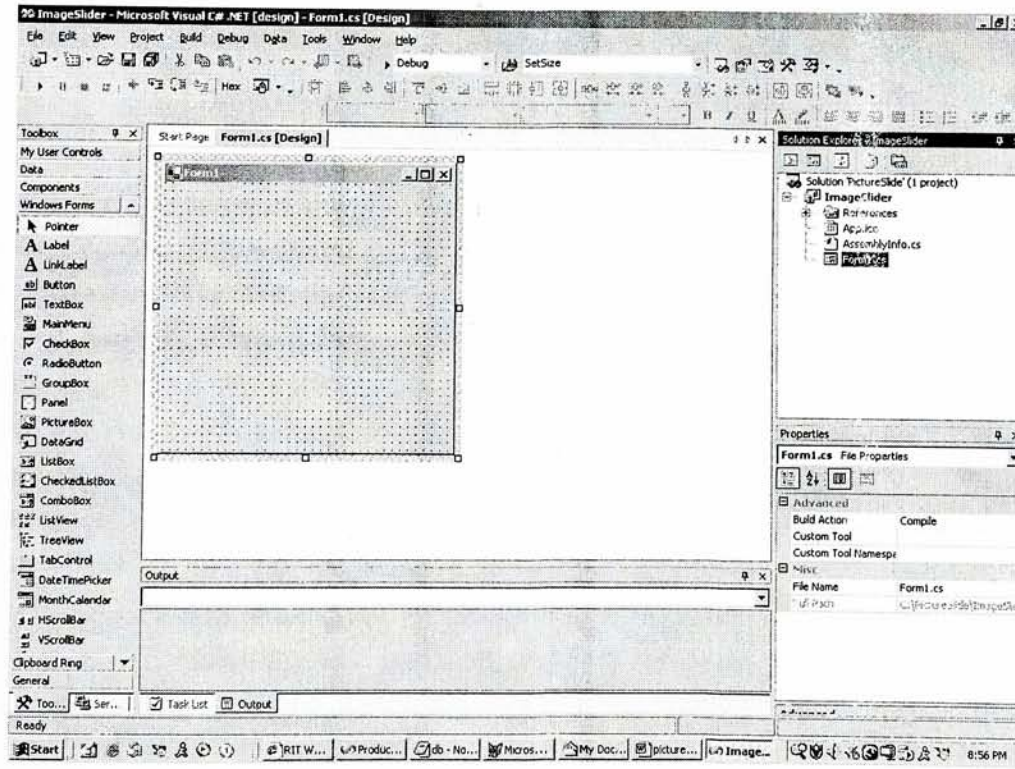

Figure a.4 – An empty Windows Form

The Basic Form:

In this section, the designer will assign properties to the form that will be used for the application.

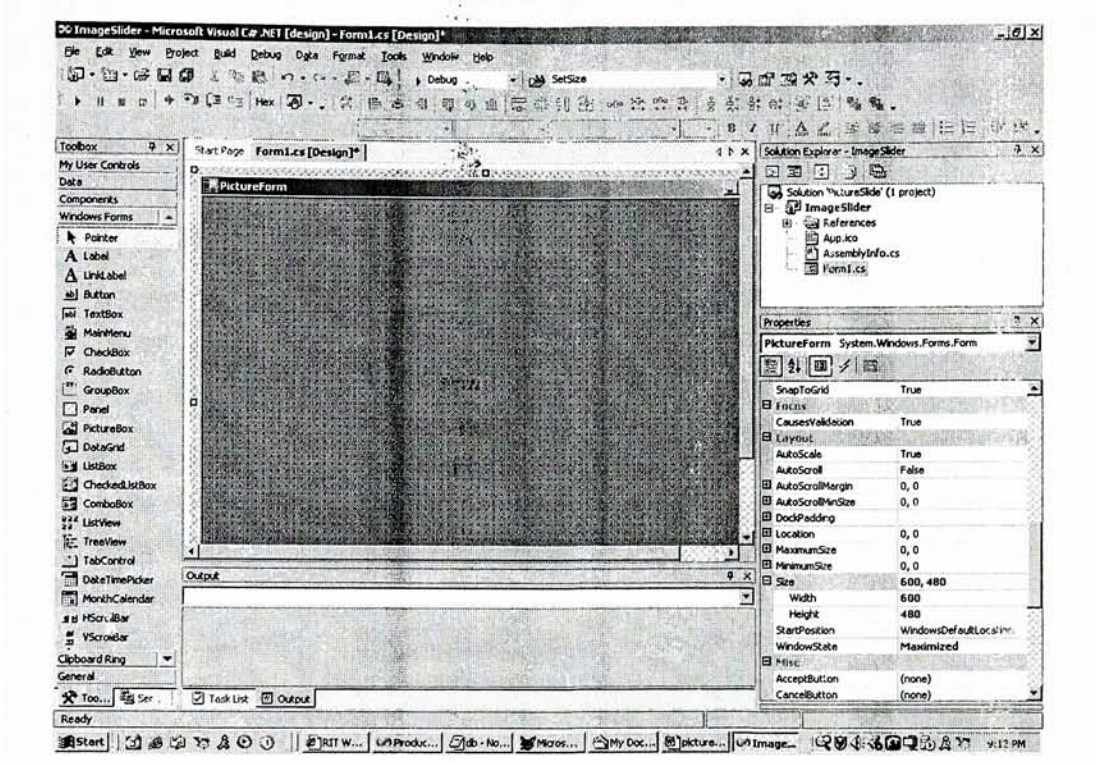

Figure a.5– Properties Window to change the properties of the form

In the Properties Window, adjust the **Size** property of the form to 600, 480 and give the form a start position at the center of the screen using the **StartPosition** property. Change the **(Name)** property of the form to PictureForm and the **Text** property to PictureForm. Notice that the heading for the form in the design view will change to Picture Form (see Figure a.5). A point to be noted is that name property is like identification and cannot contain spaces in it. Also, if desired change the **WindowState** property to Maximized. Also, set the **IsMdiContainer** property to True. All the changed properties are listed in Table a.a.

At this point, if you run the application, there will be a build error. To correct this, either hit 'F7' or right click on the form and select 'View Code' to go to the code editor.

Scroll down to the statement "Appication.Run (new Form1 ());" and change this to "Appication.Run (new PictureForm ());" and now the Application will run correctly.

(***Note**: If the Properties Window is not visible, the user can invoke it using the View menu, and selecting 'Properties Window', so also for the 'Solution Explorer' and the 'Toolbox'. These three windows have been used widely, for this application; see Figure a.6*)

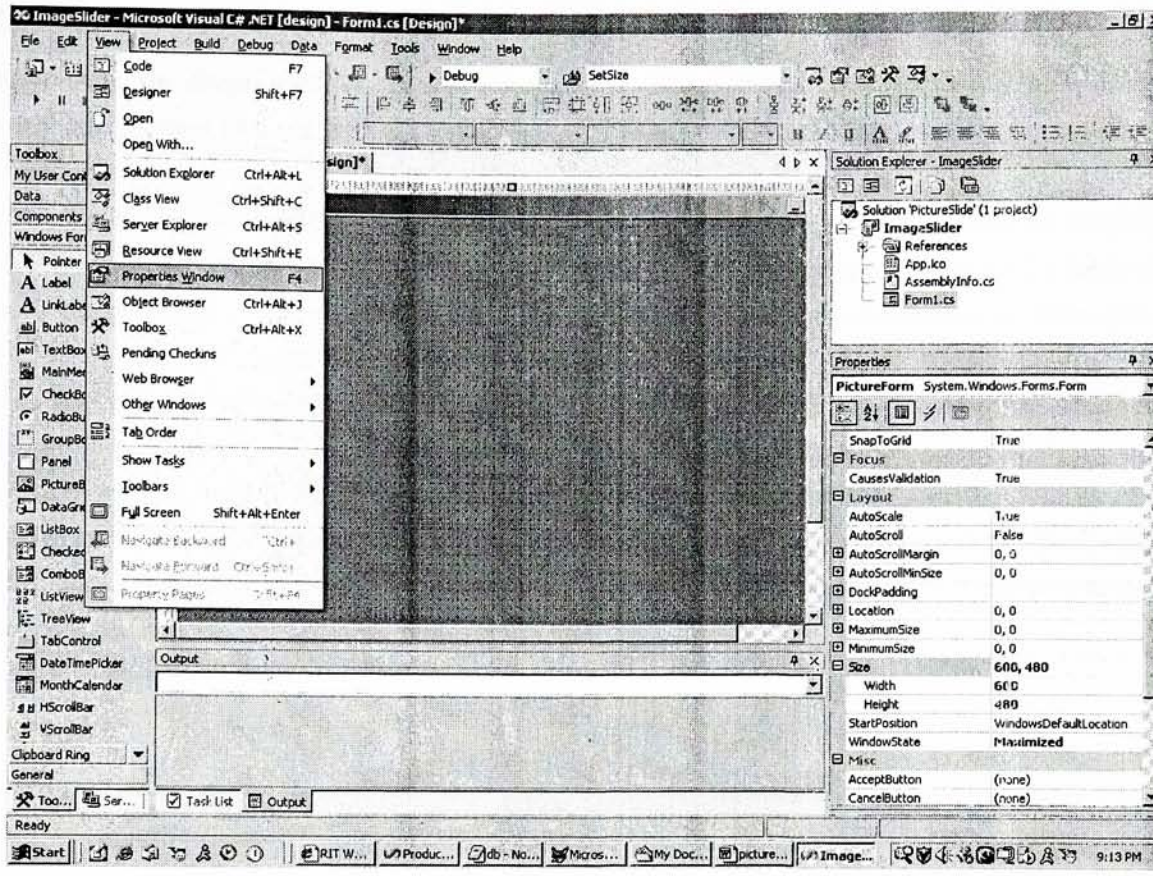

Figure a.6 – The View Pull Down menu for the Properties Window

Table a.1 – The properties for the PictureForm

Property	Value
(Name)	PictureForm
Text	PictureForm
Size	600, 400
StartPosition	CenterScreen
WindowState	Maximized
IsMdiContainer	True

The Main Menu:

In this section, the designer will create two 'Menu Items' – File and Size. The 'File' menu item will in turn have an Open, Actual Size and Exit sub-menu and the Size menu item will have 3 size options and an About sub-menu.

From the toolbox, drag a MainMenu on to the Form. This immediately creates a MainMenu item on the bottom of the IDE (space below the Design View). Change the **Name** property to mainMenu (see Figure a.7). In the design view, on the mainMenu tab, type in '&File' ('&' gives an underline to the alphabet following it). Then, below the 'File', type in '&Open', '&Actual Size', '-' and 'E&xit'. The '-' just gives a horizontal separator between two options. Also adjust the shortcuts to these tabs using the **Shortcut** property in the properties window, and change the **Name** property for each tab created to the appropriate name. For example **Name** property will be exit and the **Shortcut** property for the Exit tab will be AltF4 (see Figure a.8). Also create the other menu item Size by typing in the Tab to the side of the File menu item. Rest of the procedure to create the sub-menus is similar to the one that was just followed. The contents of these two menu items are shown in Figure a.8 and Figure a.9. The properties of the Menu Items that are created are listed in Table a.2.

Table a.2

Table a.2 – Properties for the mainMenu

Menu Item (Text)	Name	Shortcut
&File	file	CtrlF
&Open	open	CtrlO
&Actual Size	actual	CtrlA
E&xit	exit	AltF4
&Size	size	CtrlS
600 x 480	sizeChoice1	CtrlShift1
800 x 600	sizeChoice2	CtrlShift2
1024 x 768	sizeChoice3	CtrlShift3
Abo&ut	about	CtrlU

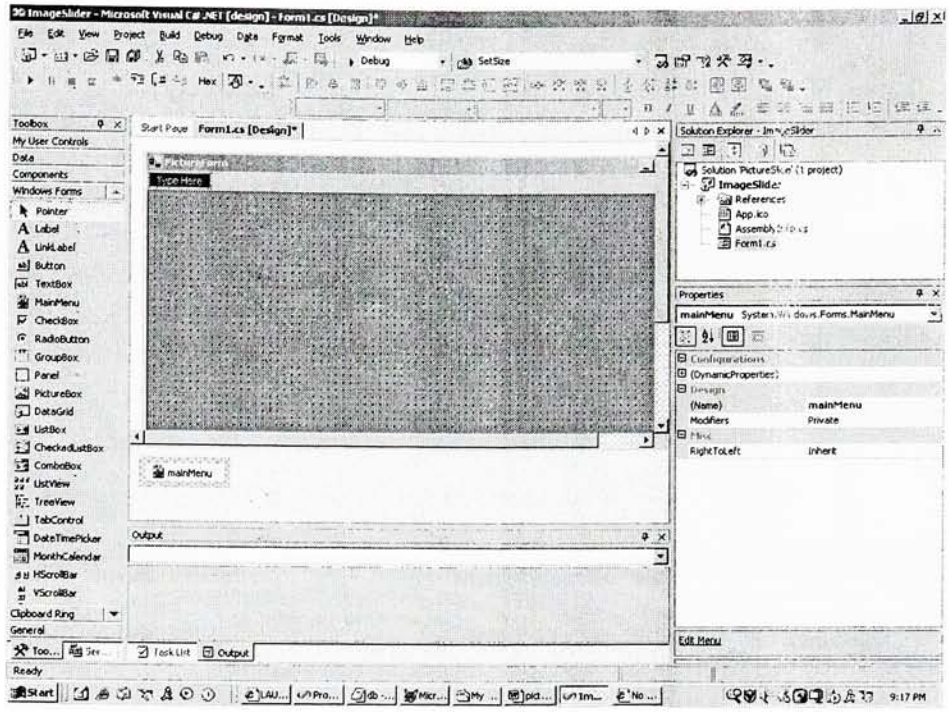

Figure a.7 – MainMenu from the Toolbar

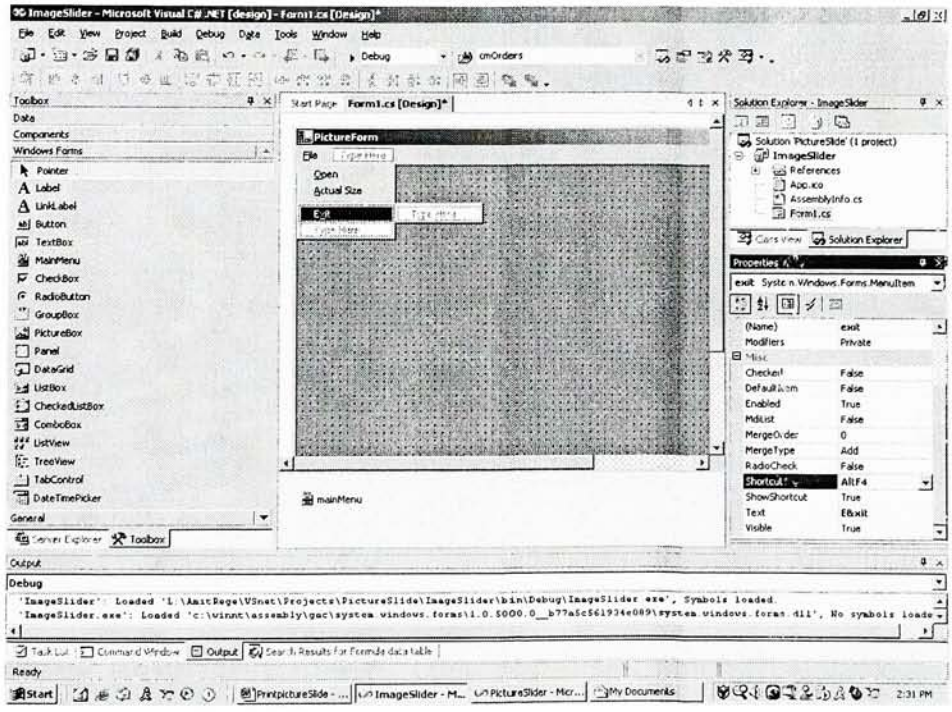

Figure a.8 – The 'File' Menu Item for the mainMenu

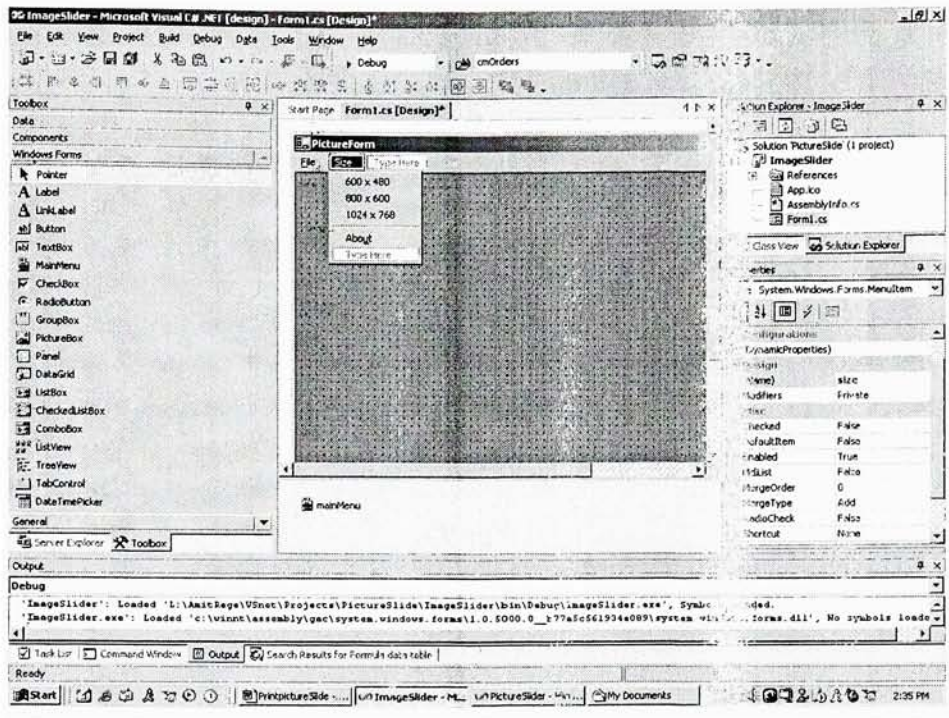

Figure a.9 – The 'Size' menu item for the mainMe

At this Point, save the Application and run it. Notice that none of the menu ms created do anything. se 'X' at the top right corner to terminate the application.

The 'ToolBar' and the 'ImageList':

In this section, the designer will create a Tool Bar with several options (buttons) on it. For this, the designer will have to create an Image List, which will hold the required images that will represent each of the Buttons on the Tool Bar. The designer will start by creating an ImageList initially followed by the creation of the ToolBar.

The ImageList:

Drag and drop an ImageList from the Toolbox on to the form. This will instantly show an imagelist icon below the design view of the form. In the properties window, change the **(Name)** property to imageList. Also, click on the **Images** property for the imagelist (see figure a.10). This will open up an **Image Collection Editor**. which can be used to add/remove the required images (see Figure a.11).

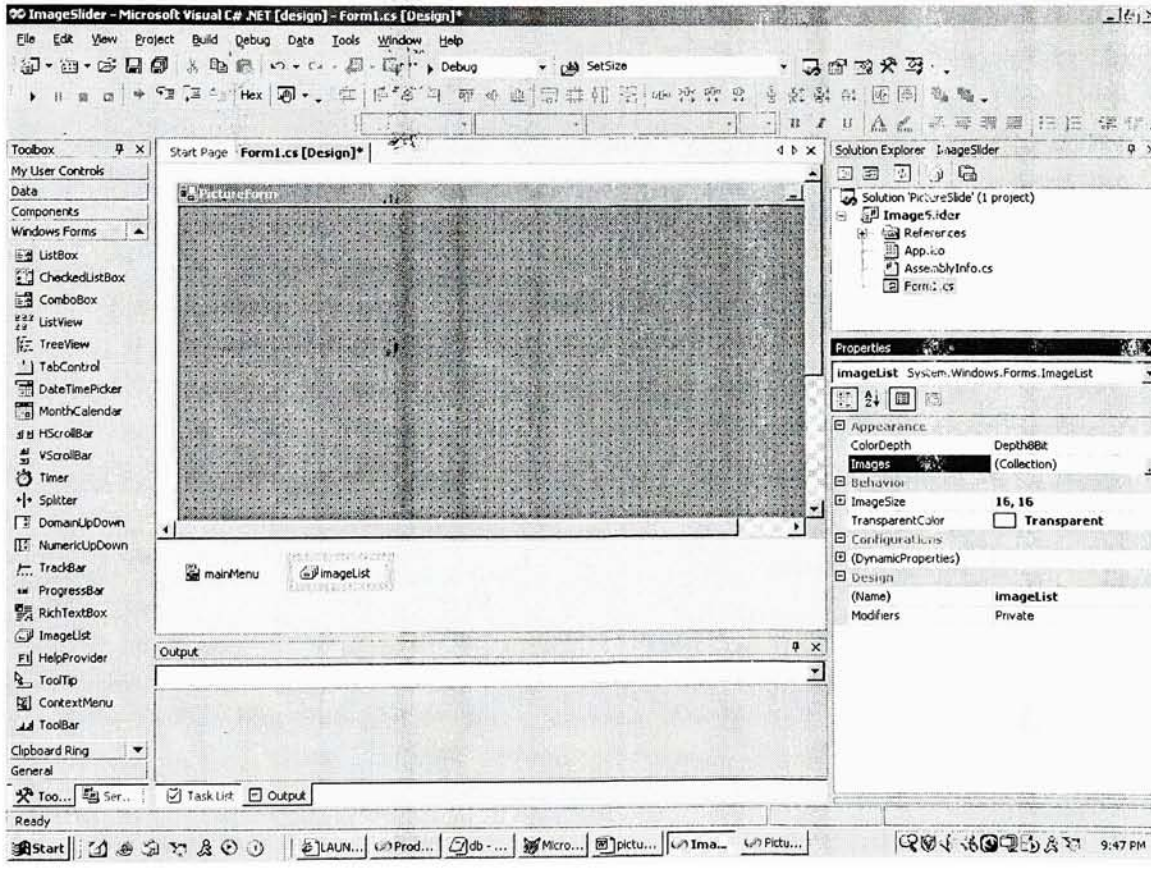

Figure a.10 – Images for the Imagelist

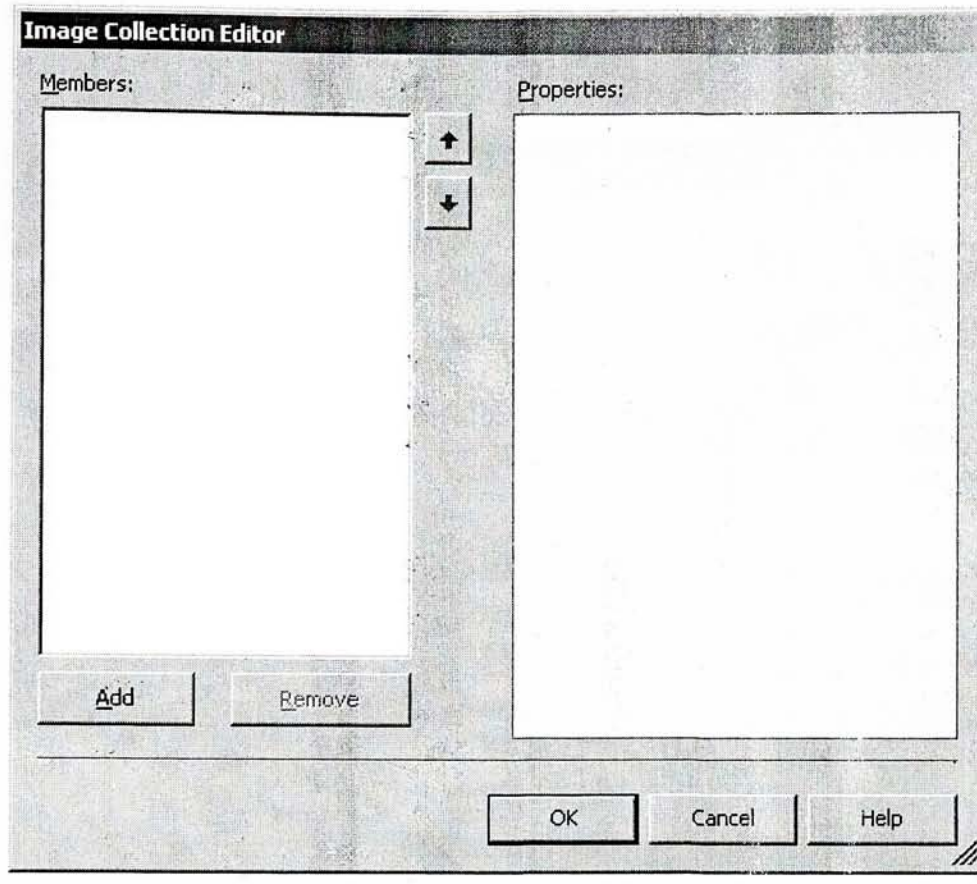

Figure a.11 – Image Collection Editor for the ImageList

One after another, add nine images to the collection editor viz: open.jpg, home.jpg, prev.jpg, next.jpg, end.jpg, zoomin.jpg, zoomout.jpg, slideshow.jpg, exit.jpg. These images get index values of '0' to '8' in that order (see Figure a.12 and Figure a.13). Hit Ok and the ImageList is ready to use.

(You may use the collection of required images provided in the folder S:\VSNET Tutorial\IconImages\)

The point to be noted is that the images that are stored in the imageList will be used as display icons for the ToolBar that will be created in the next stage of the design.

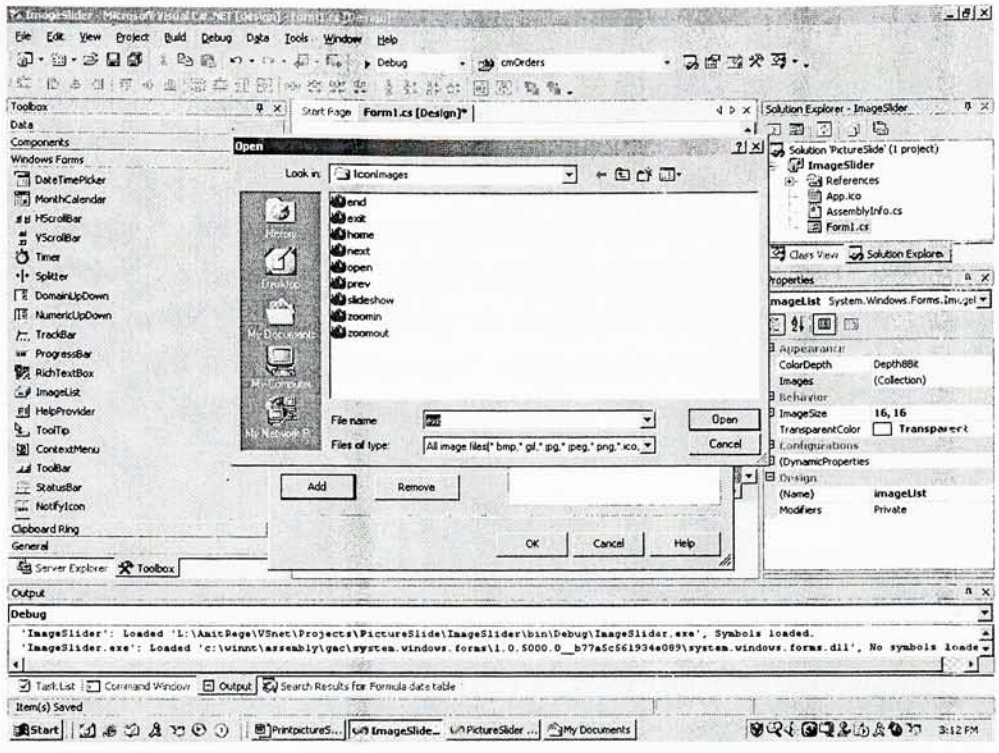

Figure a.12 – Add the appropriate images for the imagelist

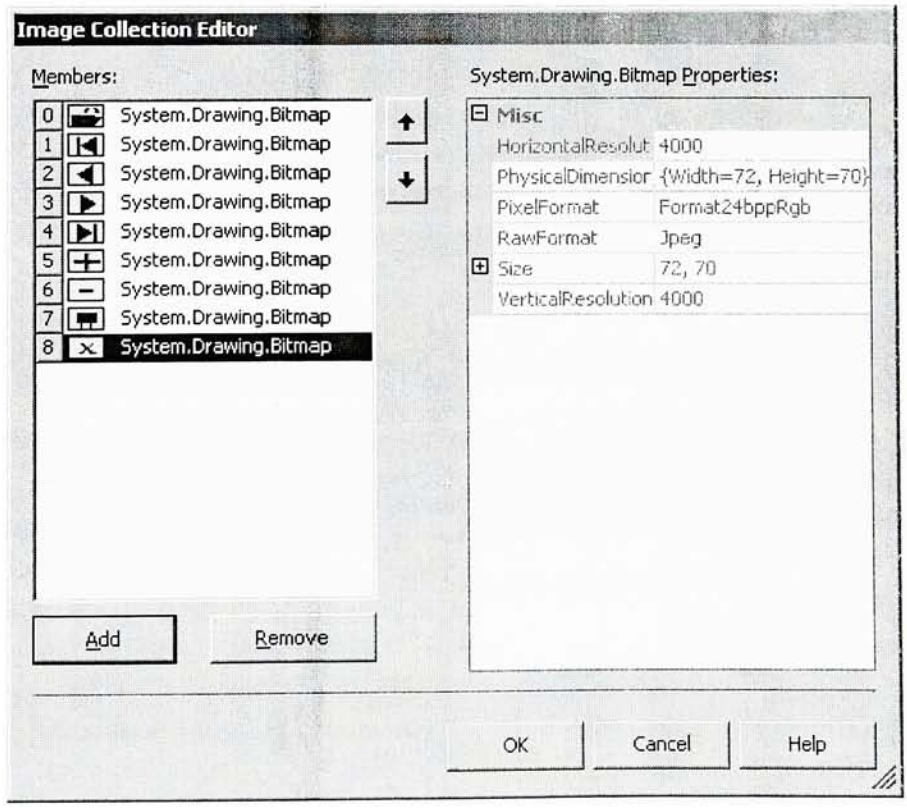

Figure a.13 – List of the selected images with the respective index values

The ToolBar:

From the Toolbox, Drag and drop a 'ToolBar' on to the form. This will appear as a horizontal line through the width of the form. In the properties window of the toolbar, change the **(Name)** property to toolBar. Change the empty **ImageList** property to imageList that is available in a drop down format (see Figure a.14). Using the Buttons property of the toolBar, open the **ToolBarButton Collection Editor** from the Properties Window. Click on Add, and add a button 'Open', change the **(Name)** property to toolBarButtonOpen and **Text** property to Open. Also, type in the appropriate text in the **ToolTipText** property. For example, for 'Open', type in Open an Image File in the **ToolTipText** property. And use the imageIndex as '0', which corresponds to the '0' on the 'ImageList'. Similarly create the 'Home', 'Prev', 'Next', 'End', 'ZoomIn', 'ZoomOut', 'SlideShow', and 'Exit' Buttons on the Toolbox (see Figure a.15 and Figure a.16). The properties for the buttons created using the ToolBarButton Collection Editor are listed in Table a.3.

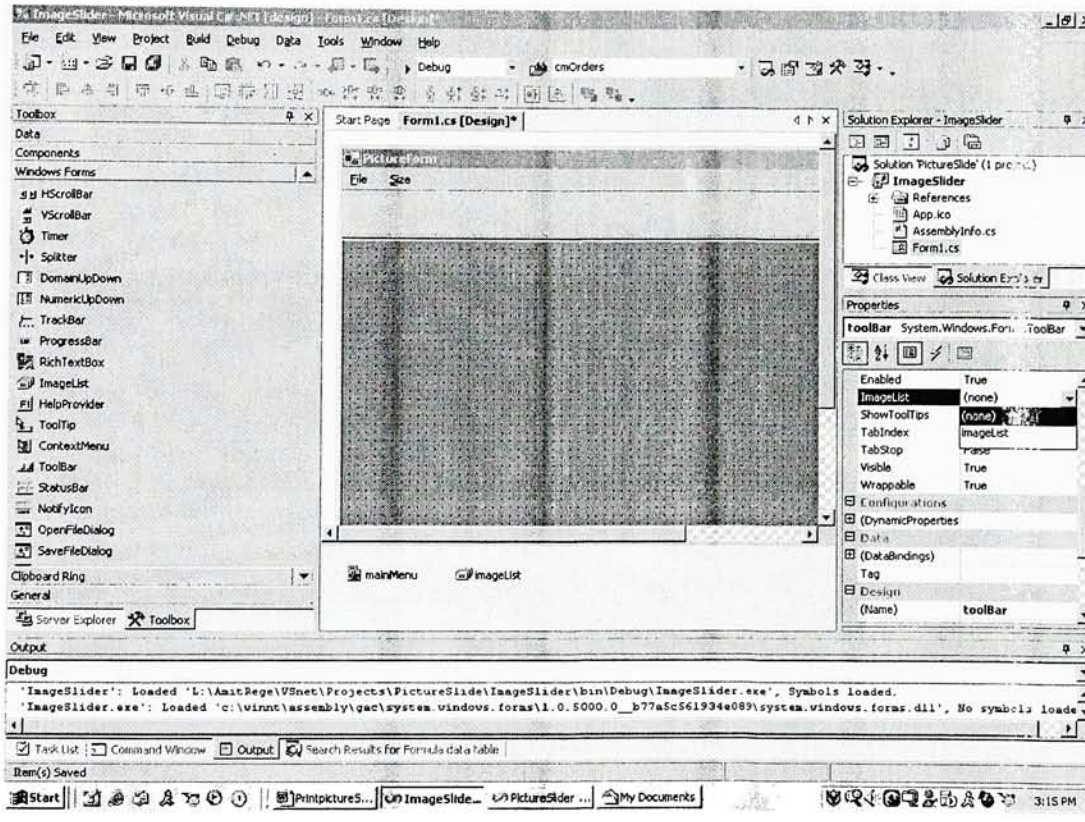

Figure a.14 – Toolbar for the application

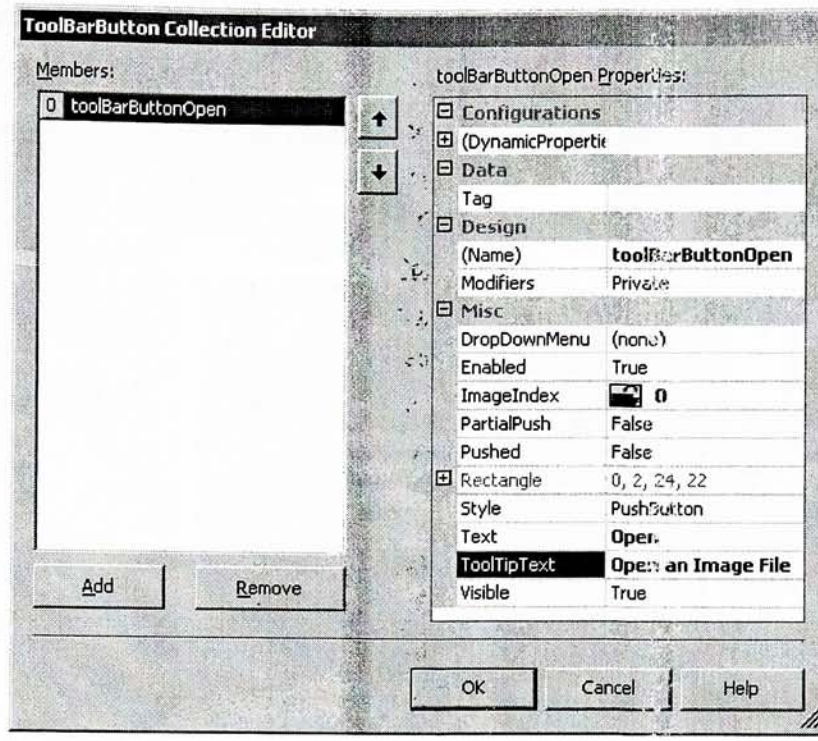

Figure a.15 – The 'open' Toolbar Button

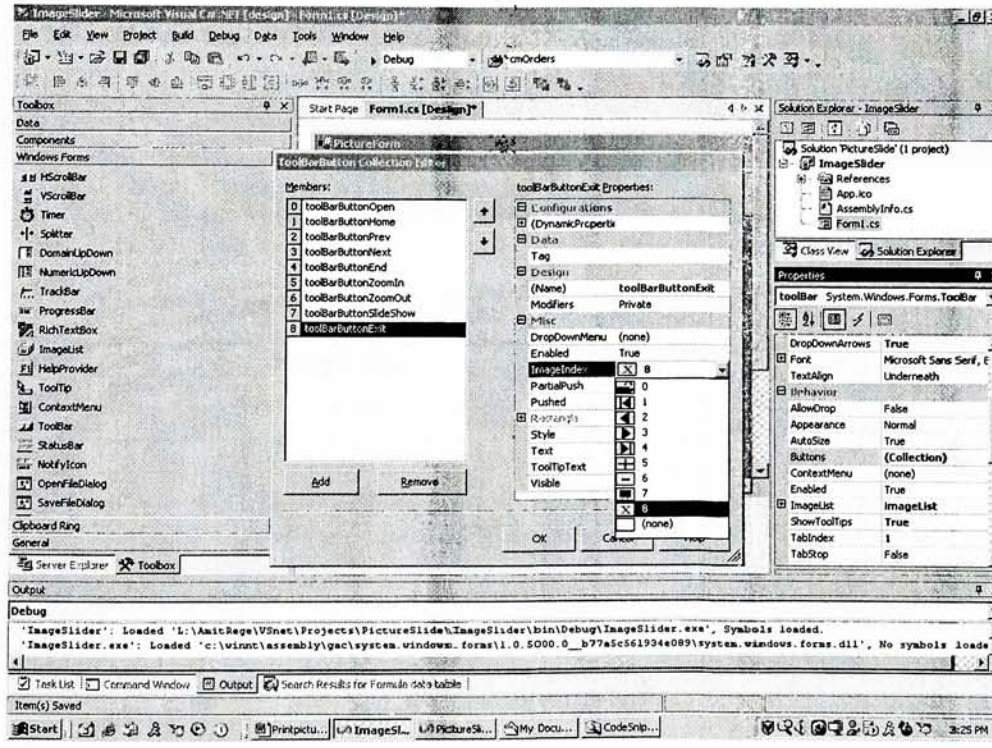

Figure a.16 – The desired buttons, images and the tips using the editor

Using the Properties Window, change the **ButtonSize** property to 60, 40 and the form in the design view will appear exactly like it is shown in Figure a.17.

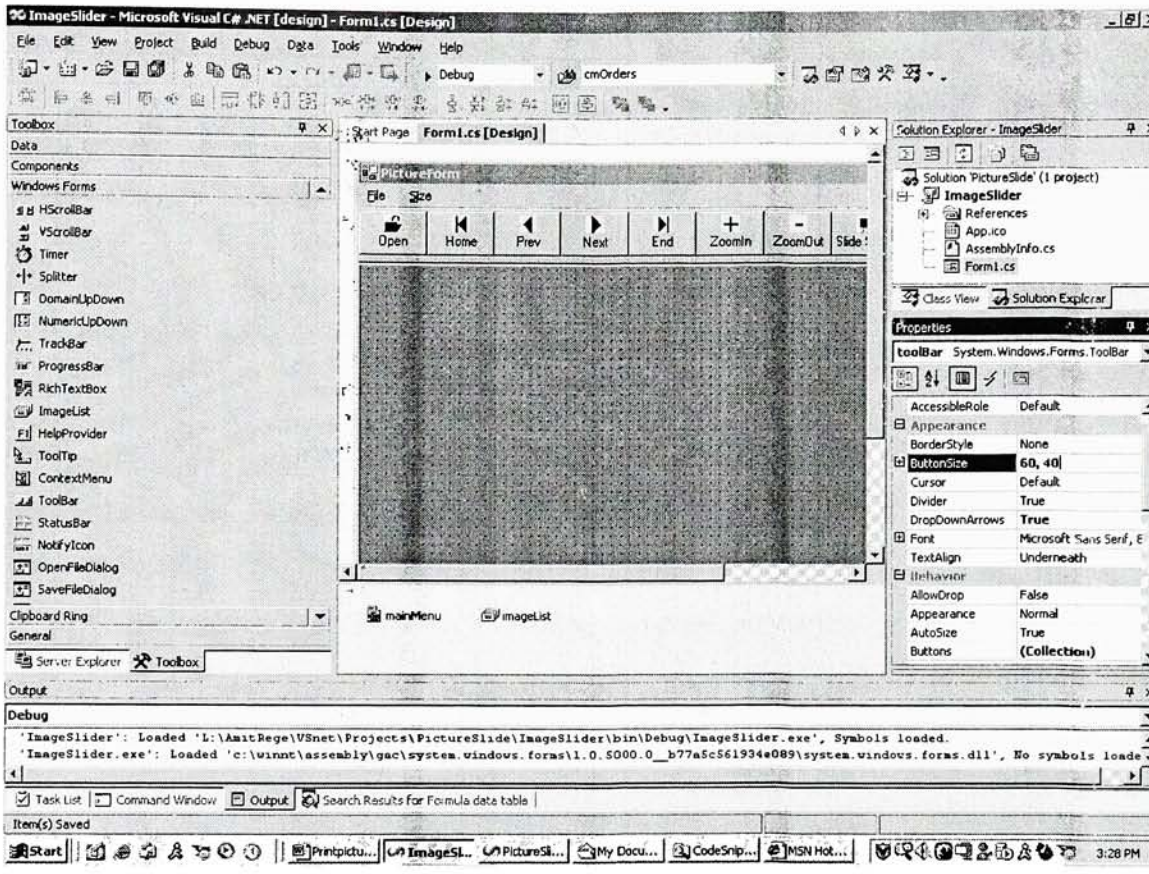

Figure a.17 – The ButtonSize Property for the toolBar

Table a.3 – Properties for the ToolBar buttons (ToolBarButton Collection Editor)

Name	ImageIndex	Text	ToolTipText
toolBarButtonOpen	0	Open	Open an Image File
toolBarButtonHome	1	Home	Go to the 1st Image
toolBarButtonPrev	2	Prev	Go to Previous Image
toolBarButtonNext	3	Next	Go to Next Image
toolBarButtonEnd	4	End	Go to the Last Image
toolBarButtonZoomIn	5	ZoomIn	Zoom In by 10 %
toolBarButtonZoomOut	6	ZoomOut	Zoom Out by 10 %
toolBarButtonSlideShow	7	Slide Show	View Slide Show
toolBarButtonExit	8	Exit	Exit the Application

The OpenFileDialog DialogBox:

In this section, the designer will create the openFileDialogBox that will be used to open the image file. During this process, the designer will set the Filter for the Image files that can be chosen, and the Initial Directory that will be opened when the user invokes the dialog box.

Drag the OpenFileDialog dialogbox from the toolbox on to the form; it immediately sets itself next to the imageList below the design view. In the Properties window, change the **(Name)** property to openFileDialog and the **Title** property to Open File. The properties such as **Filter** and **InitialDirectory** can be set for more utility (see Figure a.18).

The **Filter** property in this application is

" Image Files (JPEG, GIF, BMP etc.)|*.jpeg;*.jpg;*.gif;*.bmp|JPEG Files (*.jpeg, *.jpg)| *.jpeg;*.jpg|BMP Files (*.bmp)|*.bmp|GIF Files (*.gif)|*.gif|All Files (*.*)|*.* "

And the **InitialDirectory** property is

"S:\VSNET Tutorial\Images".

(Use the code snippet provided in "S:\VSNET Tutorial\CodeSnippets\OpenFileFilter")

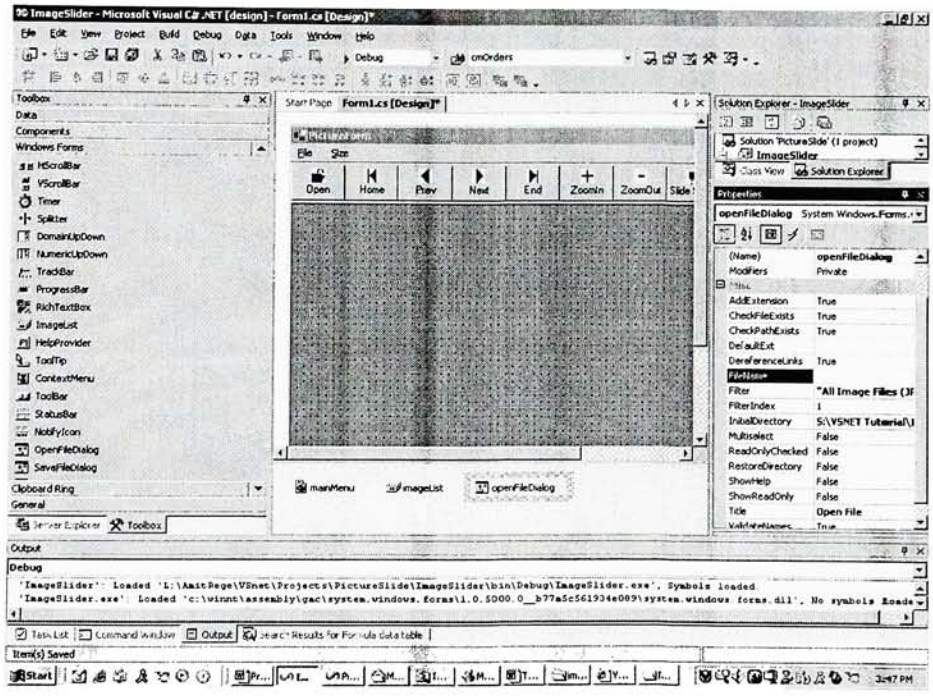

Figure a.18 – Open File Dialog Box

The Timer:

In this section, the designer will create a 'Timer', and assign a time of 1 sec between intervals for the timer. The timer is a feature that is needed in the slide show function. Drag the timer from the toolbox (This 'timer' is dragged from the 'Components' tab in the toolbox and not the normal 'Windows Forms' tab). Place it anywhere on the form. The timer then automatically places itself next to the openFileDialog below the design view (see Figure a.19).

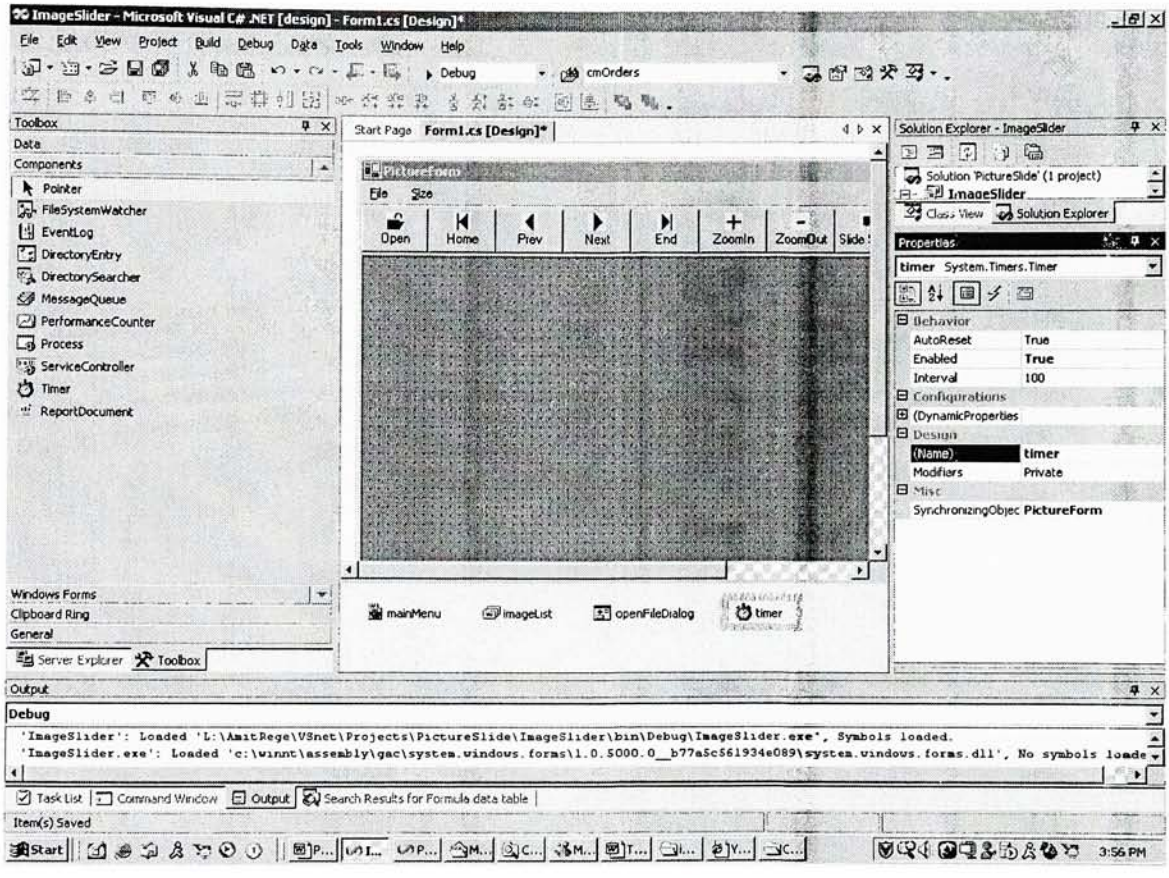

Figure a.19 – The 'Timer' from the Components **T**ab

In the properties window, set the **(Name)** property to timer and the Enabled **p**roperty to true, also set the **Interval** property to 1000 (which is one second). Also, relate this timer to the PictureForm by setting the **SynchronizingObject** property to the PictureForm, by selecting it from a pull down option (if it is already not the default).

The Status bar:

In this section, the designer will create the Status bar for the application. Here the designer will create 5 sections within the status bar to display different pieces of information like the Zoom, Mouse Position e.t.c. Coming back to the Windows Forms Tab on the Toolbox, drag the StatusBar function and drop it on to the form. Change the **(Name)** property to statusBar and delete any text in the **Text** property, and the **ShowPanels** to True (see Figure a.20). Click on the **Panels** property to add different sections within the status bar using the **StatusBarPanel Collection Editor** (see Figure a.21).

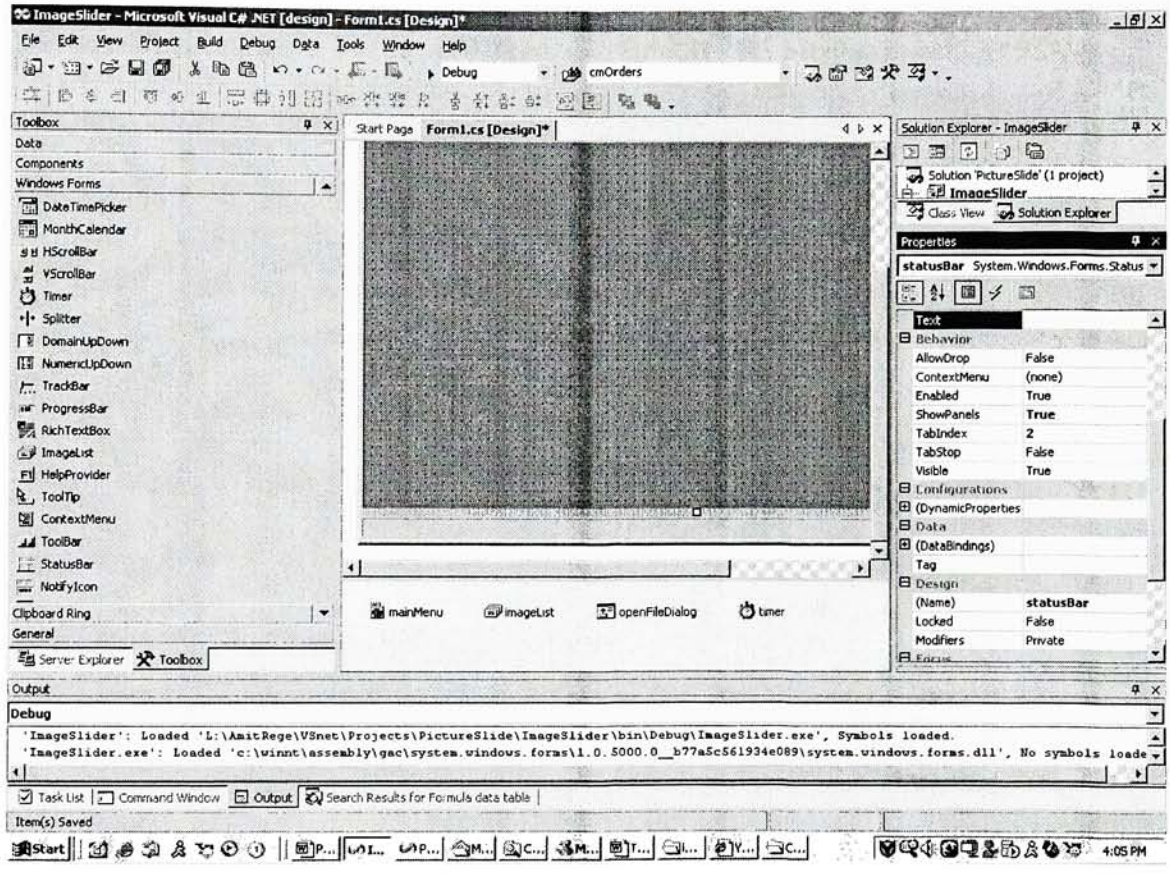

Figure a.20 – The Status Bar

Using the Panel Collection Editor, add the following five tabs

1. Status – To show the current status of the application.
2. Mouse – To show current Mouse Position.
3. File – To show the file number corresponding to the image file on display.
4. Size – To show the size of the Image.
5. Zoom – To show the Zoom of the Image.

Set the **(Name)** property of the added sections to appropriate names. For example, the name for the section for mouse position will be statusBarPanelMouse (see Figure a.21). Properties for the different sections of the Status Bar are listed in Table a.4.

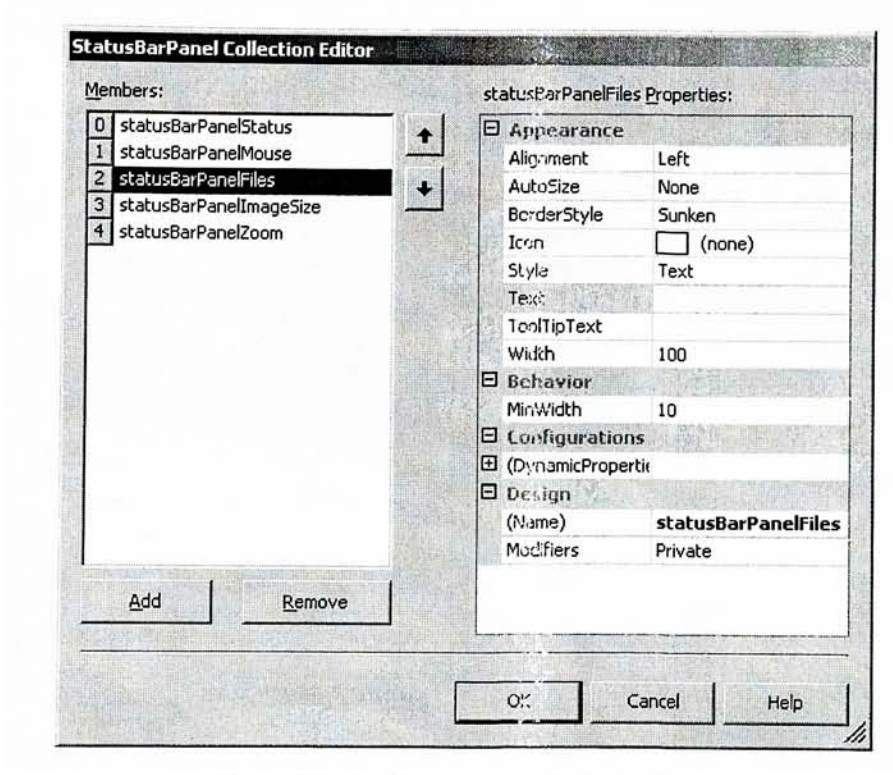

Figure a.21 – The StatusBarPanel Collection Editor

Table a.4 – Properties for the StatusBar sections

Index Number	Name
0	statusBarPanelStatus
1	statusBarPanelMouse
2	statusBarPanelFiles
3	statusBarPanelImageSize
4	statusBarPanelZoom

The About Box:

In this section, the designer creates another Form called the About Box;about this is just one way of displaying information about the application and the person who created the application. It also is a good way of getting accustomed to a MDI application; where more than one form is used in the same application.

In the solutions explorer, select the project 'ImageSlider' and add a new form to it, by using the right click and selecting the 'Add Windows Form' tab (see Figure a.22).

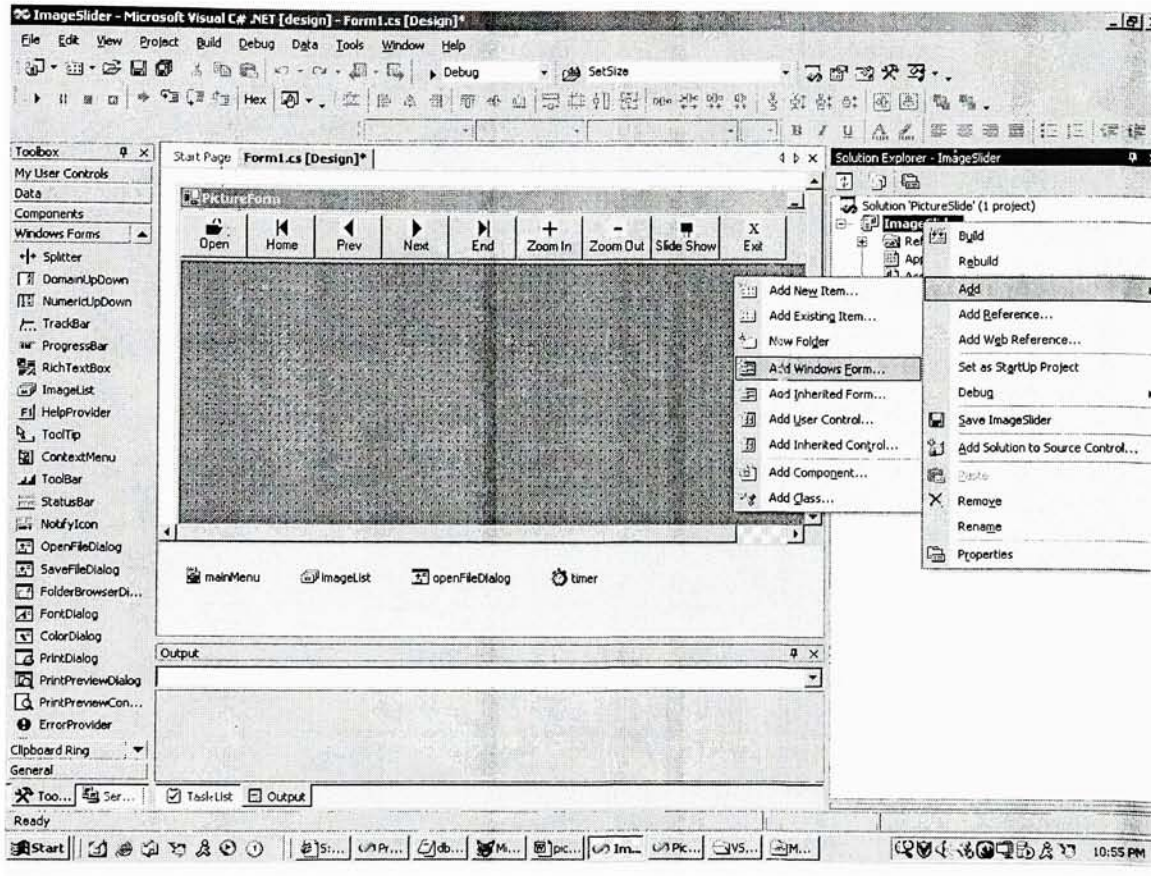

Figure a.22 – A second form for the 'About Box'

Change the **(Name)** property to about and **Text** property to About Box, using the properties window. Drag a **RichTextBox** in the new form for the About Box, and fill the box with the desired text using the **Text** property, and save the project (see Figure a.23).

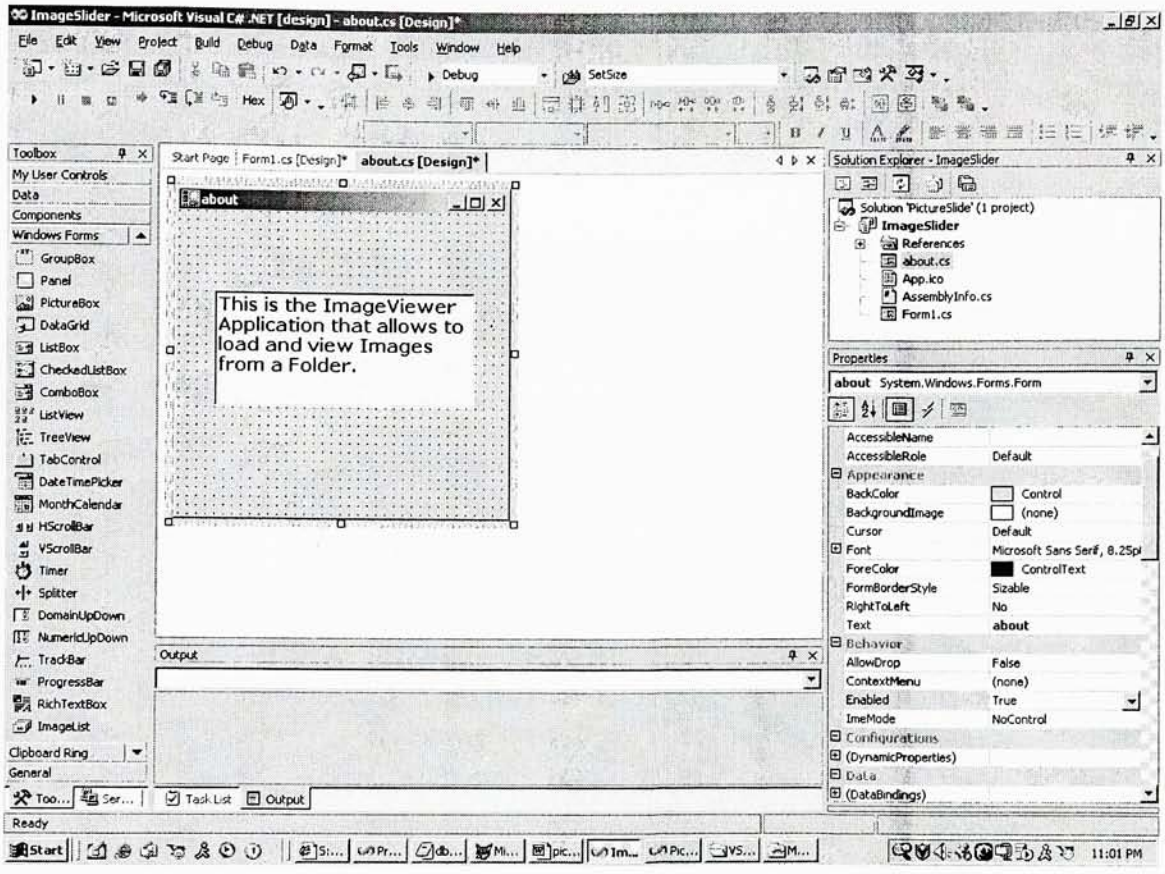

Figure a.23 – The RichTextBox for the About Form

Using the Solutions explorer return to the PictureForm, which was designed earlier. Now that the designs of the forms are complete, it is time to add some functionality to the form and the visual aids that have been created.

Event Handling for toolBar Buttons:

In this section, the designer inputs some code to add functionality to the tool bar that was created.

Click on the Toolbar and select the events function in the properties window. In the behavior **ButtonClick**, input a name onToolBarClick and hit enter. This opens up the Code editor and the focus is between the parentheses of the Function 'OnToolBarClicked' (see Figure a.24 and a.25).

Copy and Paste the code from the code snippet present at the location "S:\VSNET Tutorial\CodeSnippets\toolBarButtons.txt"

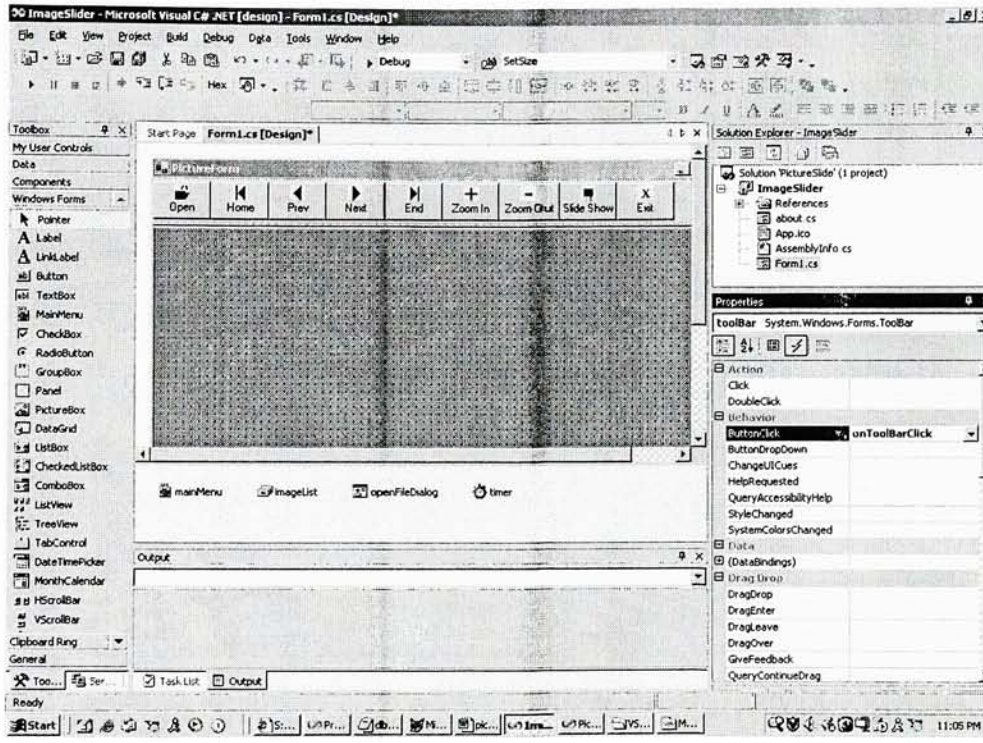

Figure a.24 – Event handling for the toolbar

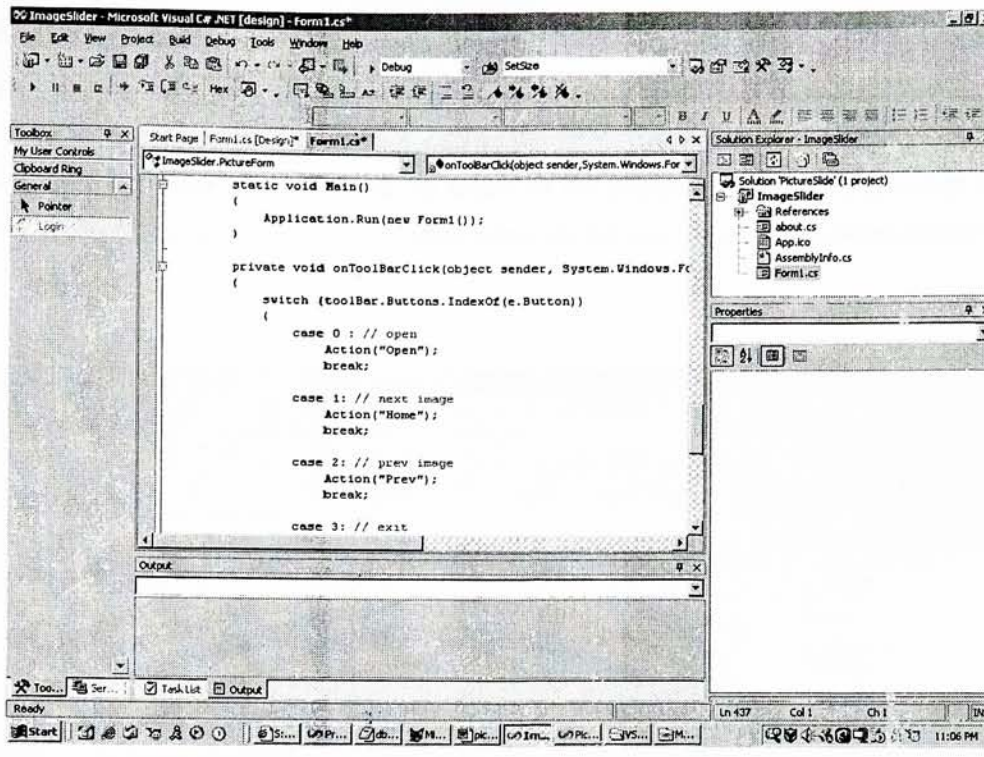

Figure a.25 – Code editor for the toolbar 'onclick' event

Event Handling for mainMenu:

In this section, the designer adds functionality to the main menu items that were created. Now click on the mainMenu icon that is placed below the design view of the form. This invokes the mainMenu that was created earlier. Select the Open sub-menu and choose the events in the properties window. In the event **Click** input a function name onOpenClick and hit enter. This again opens up the code editor at the correct position (see Figure a.26).

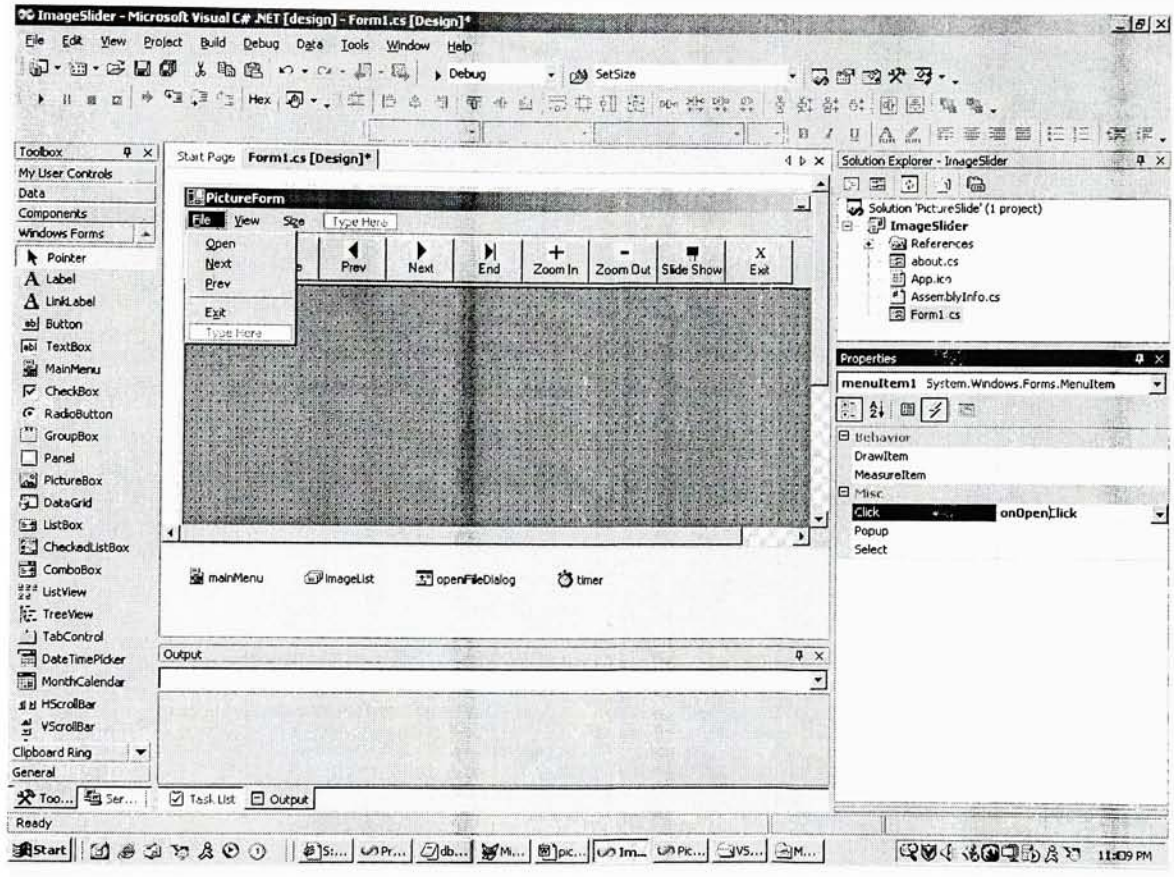

Figure a.26 – Event Handling from the properties window

Type in the following code:

```
Action ("Open");
```

Follow similar steps for the rest of the sub-menus on both the Menu Items, the code for these sub-menus is listed in Table a.5.

Table a.5 – The code for the sub menus

Sub-menu	Click-event name	Code
Open	onOpenClick	Action("Open");
Actual Size	onActualSizeClick	Action("ActualSize");
Exit	onExitClick	Action("Exit");
600 x 800	onsizeChoice1	Action("Size1");
800 x 600	onsizeChoice2	Action("Size2");
1024 x 768	onsizeChoice3	Action("Size3");
About	onAboutClicked	Action("About");

(Note: The Function 'Action' is described later)

Event Handling for the Form:

In this section, the designer adds functionality to the Paint event of the form, which is used to display the selected image. Also another two events are given some functionality, these events being 'Form Load' and 'Mouse Move'.

Click on the events tab for the form in the properties Window, and fill out the Paint, Load and MouseMove events and fill in the required code for them. Once the name for the function is typed in and the Enter key is pressed the code editor opens at the appropriate place (see Figure a.27). The name of the click event and the location for the code are listed in the Table a.6.

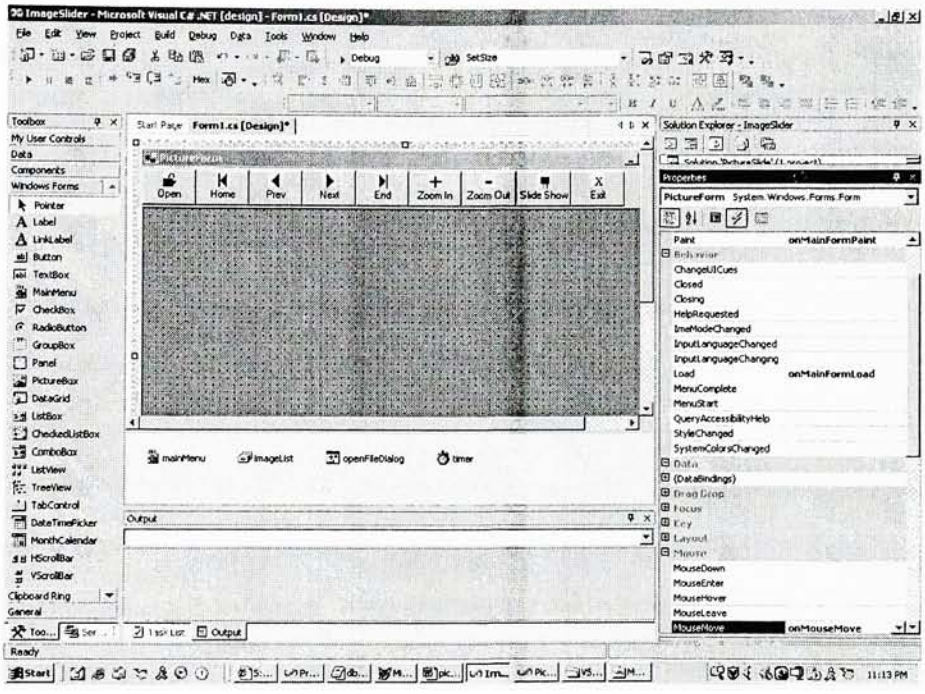

Figure a.27 – Events for the PictureForm

Table a.6 – Location of the event property code for PictureForm

Event Property	Click Event Name	Location for the Code
Paint	onMainFormPaint	S:\VSNET Tutorial\CodeSnippets\PaintHandlerFunc.txt
MouseMove	onMouseMove	S:\VSNET Tutorial\CodeSnippets\MouseMoveFunc.txt
Load	onMainFormLoad	S:\VSNET Tutorial\CodeSnippets\FormLoadFunc.txt

In this section, the designer defines some variables that are used in providing functionality to the application.

For this application following are the user-defined variables:

(**Private**: The private keyword is a member access modifier. Private access is the least permissive access level. Private members are accessible only within the body of the class or the struct in which they are declared.

Public: The **public** keyword is an access modifier for types and type members. Public access is the most permissive access level. There are no restrictions on accessing public members.)

```
private int FileIndex = -1;                    //Index number for the file that is to be opened
private string FilePath;                       //Path of the file that needs to be opened
private ArrayList FileArray;                   //Array to store all the files that can be opened
private ArrayList FileTypes;                   //Array to store the types of files that can be opened
private Bitmap MyBitmap;                       //Object to open the current image
private Size MyBitmapSize = new Size(640, 480); //Size for the MyBitmap
private Size MyFormSize   = new Size(0, 0);    //object to store a temp size structure for resizing the form
private PointF MyPoint = new Point(320, 240);  //object to store the position where the image will be drawn
private float fScale = a.00F;                  //amount to which the image is to be scaled
private int timebetweenfiles = 1000;           //value of the time interval between slides
private int sizeFitIndex = 0;                  //whether to resize the image to fit window
```

Also for this Application, the following two namespaces have to be used. The designer can directly type this in the code editor below the space where the other classes are defined.

```
using System.Drawing.Drawing2D;                    //Display Images
using System.IO;                                   //Directory Assistance
```

Variables can either be directly typed inside the code editor or can be declared using the IDE by using the class **view** (see Figure a.28 and a.29).

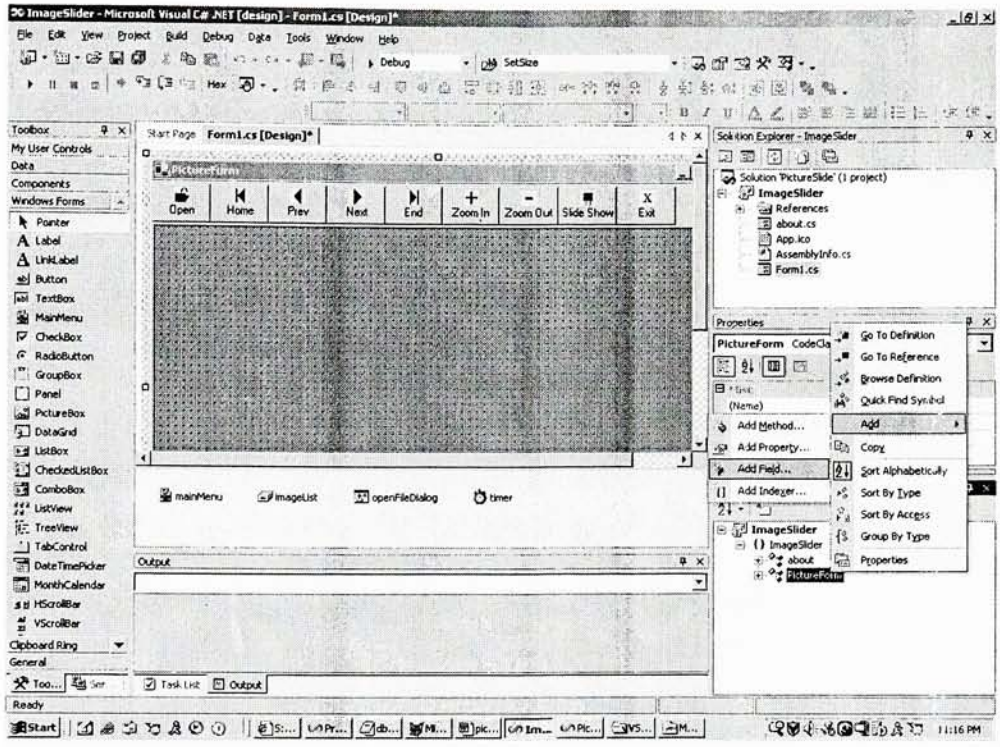

Figure a.28 – Variable declaration using the class view

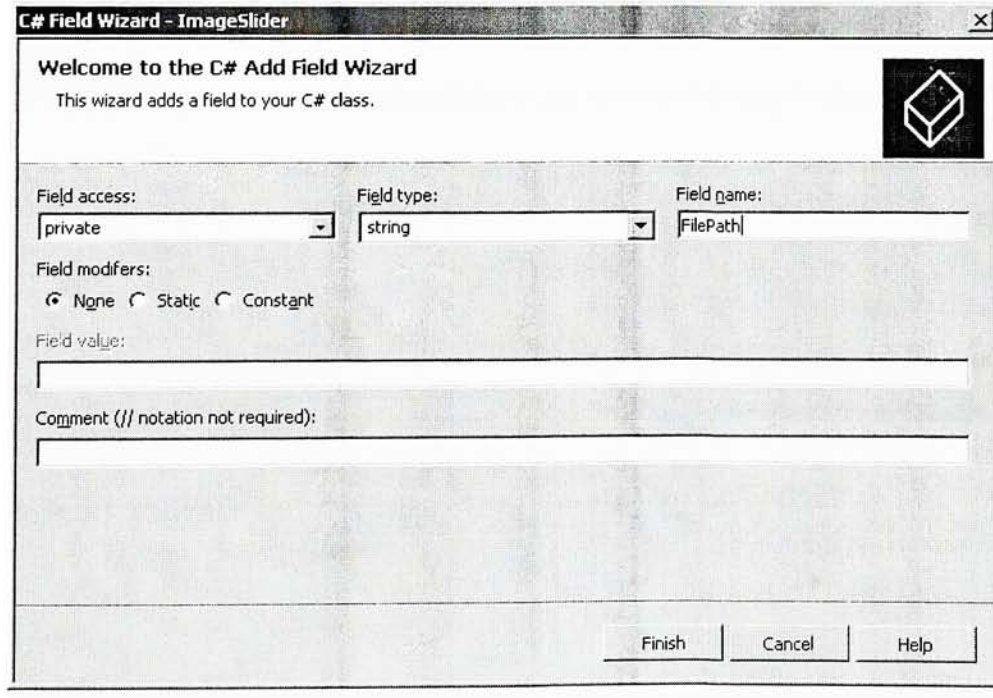

Figure a.29 – Using the field Wizard from the Class View

Event Handling for the SlideShow (Timer):

In this section, the designer adds the required code for the slide show.

Click on the events tab for the timer from the properties window, in the event **Elapsed**, type onTimerElapsed and hit enter. This opens the code editor; enter the code at the following location: "S:\VSNET Tutorial\CodeSnippets\TimerFunc.txt".

Functions used to provide Functionality to the Application:

In this section the designer provides the code in which the functions used for the applications are defined.

In a similar manner, by using the class view wizard or directly typing into the code editor the following functions are defined to add functionality to the application.

These functions are:

1. GetFilepath( ) -- "S:\VSNET Tutorial\CodeSnippets\GetFilePathFunc.txt"
2. SetPicture( ) -- "S:\VSNET Tutorial\CodeSnippets\SetPictureFunc.txt"
3. ProcessDirectory( ) -- "S:\VSNET Tutorial\CodeSnippets\ProcessDirectory.txt"
4. Action(String dummy) -- "S:\VSNET Tutorial\CodeSnippets\ActionFunc.txt"

The Image View Application is now complete. The designer can now build the application and run it to see it work.

Reference:

1. Rakesh Rajan, "How to Load/Display images with C#", The Code Project, 24th Feb 2003
http://www.codeproject.com/cs/media/quickview.asp?target=Image%7CViewer

Appendix B: Database Normalization

Normalization Overview

A database table is a collection of instances of data that have the same general attributes. The data in a database table is arranged in rows and columns. This data represents information relating to entities (real-world concepts). Each entity has one or more attributes. Entities in different tables can have relationships between them that help explain a situation better. A database can be defined as a collection of logically organized, related information stored in data tables that serve a specific purpose.

In the year 1969, IBM researcher, Dr. E. F Codd, developed the concept of normalization. Codd described a database as a collection of unordered tables that can be manipulated using non-procedural techniques that would return tables. The key concept was that by using the relationships the designer would be able to create virtual tables to meet specific needs [1]. Normalization can be defined as "a process of systematically organizing data in a database" The principle goals of normalization are to avoid redundancy of the data and make certain that only related information is stored in a table. This insures that the database will consume the minimum space and work at its most efficient. The database community came up with specific guidelines to follow in order to make a database normalized. These guidelines are referred to as "normal forms". The normal forms range from the 1st normal form to the 6th normal form. These normal forms are additive. This means that a database in the 3rd Normal form must and should be or will be in the 2nd normal form. The normal forms are referred by the letters NF. Thus, they range from 1 NF to 6NF. In most of the practical application, the normalization to the 3NF is good enough.

b.1 The First Normal Form (1NF)

The 1NF essentially aims at eliminating the repetitive groups of data. By designing a table to the 1st normal form, the elements of data that need to be stored are identified and some level or organization is required. For example, in the customer table of the MESASI database, we know each customer may have one or more customer orders. Thus, if the customer order number or numbers are included in the customer table then the structure would need to be like shown in table b.1

Table b.1 - Table that needs to be in 1NF

Customer Name	Order number 1	Order number 2	Order number x
Xerox Corporation	1212154	2222222	2121212
Bose Corporation	1215458		
TRW automotive	4545454	5454545	

Thus, it can be seen that the order number '2' to order number 'x' is an empty field in the table and consumes space. This can be structured more appropriately by designing another table that holds all information about the orders, and then relates each order to a customer using a unique identifying key for each customer. Thus, the order table can have the elements shown in table b.2

Table b.2 - Order table in 1NF

Order number	Customer Id	Items count	Amount
1212154	4444444	2	1500
1215458	5555555	1	2500
4545454	8888888	4	4000

The unique identifier of each table is called the primary key. It is unique to each of the entities in the table. Thus, the 1NF helps in eliminating repeating groups of data, creating a separate table for each of the entity types and identifying the primary keys.

b.2 The Second Normal Form (2NF)

The 2NF attempts to eliminate the functional dependencies within a table. Functional dependency identifies whether a column of a table is dependent on another column. To clearly distinguished this point, if the primary key of a table is a composite key, then the table should contain columns that are the attributes of both the elements that form the composite key and if they represent a attribute that is linked to just one of the elements of the composite key, there is a possibility of repetition of datb. In the MESASI, the 2NF can be explained using the table b.3.

Table b.3 - Table that needs a 2NF conversion

Order number	Customer Id	Item number	Items count	Date
1212154	4444444	1	2	01/01/2001
1212154	4444444	2	2	01/01/2001
1215458	5555555	1	1	02/01/2002
4545454	8888888	1	4	05/15/2003
4545454	8888888	2	4	05/15/2003
4545454	8888888	3	4	05/15/2003
4545454	8888888	4	4	05/15/2003

Thus, it can be seen that the information is very repetitive and assuming that the primary key for the table in b.3 is a combination key of the 'item number' and 'order number', not all the columns are dependent on both the factors that define the composite primary key. In order to reach a 2NF form, separate tables must be created for the order. These two tables are called as 'customerorderdetail' and 'customerorderheader'. These tables can be represented by tables b.4 and b.b. Thus, the information is broken down into more than one table if required and the database gets a more efficient structure.

Table b.4 - customerorderheader table

Order number	Customer Id	Date	Items count
1212154	4444444	01/01/2001	2
1215458	5555555	02/01/2002	1
4545454	8888888	05/15/2003	4

Table b.5 - customerorderdetail table

Item number	Order number	Part	Quantity
1	1212154	Screw	200
2	1212154	Tmale	500
1	1215458	Cube	200

b.3 The Third Normal Form (3NF)

The third normal form further entails the elimination of the functional dependencies within a table. This tries to identify the data column that does not directly relate or give information about the primary key of the table. This can be seen in table b.b.

Table b.6 - Order table that needs a 3NF conversion

Order number	Customer Id	Customer Name	Items count
1212154	4444444	Xerox Corporation	2
1215458	5555555	Bose Corporation	1
4545454	8888888	TRW automotive	4

Thus, it can be seen that 'Customer Name' is dependent on 'Customer Id', thus it makes good business sense to accommodate the data relating to the customer in a separate table and link each order to the customer by using the 'Customer Id'. The 3NF provides the ideal example of what is called a 'foreign key'. This is the key that is a primary key of another table and is included as a component column in a different table. In the table b.7, the foreign key is the 'Customer Id' and that is the primary key of the customer table represented by table b.8

Table b.7 - Order Table

Order number	Customer Id	Date	Items count	Amount
1212154	4444444	01/01/2001	2	1500
1215458	5555555	02/01/2002	1	2500
4545454	8888888	05/15/2003	4	4000

Table b.8 - Customer Table

Customer Id	Customer Name	Join Date	Tel	Fax
4444444	Xerox Corporation	01/10/2000	5852854444	5852854442
5555555	Bose Corporation	02/01/2001	5858888888	5858888882
8888888	TRW automotive	05/15/2002	3155558585	3155558582

b.4 Higher Forms of Normalization

For most of the practical purposes the 3NF form is sufficient, there are some rare cases in which further normalization is required. These problems usually involve tables with three or more columns and all those columns are keys.

These advanced forms of normalization are as follows:

- The Boyce-Codd Form (BCNF)
- The Fourth Normal Form (4NF)
- The Fifth Normal Form (5NF)
- The Sixth Normal Form (6NF)

b.4.1 The Boyce-Codd Form (BCNF)

This is just a more rigorous form of the 3NF. It involves multiple overlapping candidate keys [2]. In this situation, all the candidate keys are composite keys, and there is more than one candidate key. Also, the candidate keys are overlapping such that each has at least one column that is common with another candidate key.

b.4.2 The Fourth Normal Form (4NF)

A relation is said to be in fourth normal form if each table contains no more than one multi-valued dependency per key attribute. Assume an employee can be assigned to multiple projects and also has more than one skill, thus all three attributes (employee number, skill and project) need to form the key to identify a specific instance [3]. Here there is multi-valued dependency between the employee number and the skill and also between employee number and project. To convert these into a 4NF both pairs of the multi-valued dependencies are placed in new tables.

b.4.3 The Fifth Normal Form (5NF)

The 5NF indicates when an entity cannot be further decomposed [3]. The aim of fifth normal form is to have relations that cannot be decomposed further. A relation in 5NF cannot be constructed from several smaller relations. Thus, a table would be in 5NF if all columns are candidate keys and that the tables' primary key does not consist of more than one column.

b.4.4 The Sixth Normal Form (6NF)

The 6NF is extremely difficult to achieve in a real world environment. When a system is in 6NF, changing data in one place automatically prompts the change at all other necessary places such that the entire system is in-sync with the change.

Reference:

1. Mike Chapple, *"Database Normalization Basics"*, Databases Blog, About Inc. A PRIMEDIA Company, 14th March, 2004.
   http://databases.about.com/library/weekly/aa080501b.htm
2. Robert Vieira, *Professional SQL Server 2000 Programming*, Wrox *Press.*, 2003.
3. Information Technology Services, *"Introduction to data modeling"*, The University at Texas Austin Company, 29th February, 2004.
   http://www.utexas.edu/its/windows/database/datamodeling/rm/rm8.html