

OPTIMAL FERTILIZER INVESTMENT SYSTEM

A COM – BASED APPLICATION

By

LI WANG

Bachelor of Arts

Beijing Polytechnic University


Beijing, People's Republic of China

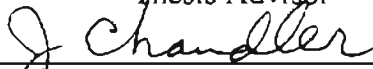
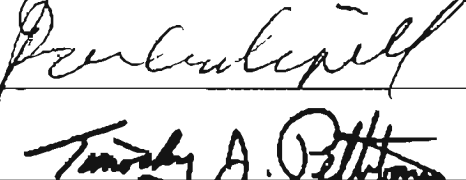
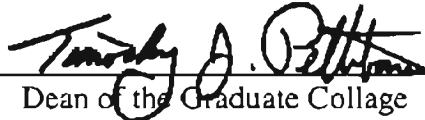
1983

Submitted to the faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 2001

OPTIMAL FERTILIZER INVESTMENT SYSTEM  
A COM – BASED APPLICATION

Thesis Approved:

  
\_\_\_\_\_  
Thesis Advisor

  
\_\_\_\_\_  
  
\_\_\_\_\_  
  
\_\_\_\_\_  
Dean of the Graduate Collage

## ACKNOWLEDGMENTS

This thesis is the outcome of many months of my work. I am very grateful to the individuals who, directly or indirectly, have helped me in my effort. First, I wish to thank my advisor, Dr. G. E. Hedrick. His erudite knowledge, constructive guidance and great insight on the relationship among the applications have contributed much to the ability to develop an integrate software with modern technology; his tirelessness and meticulousness in capturing and editing my manuscript have helped me to set the overall tone and style of the thesis and made the inevitable revisions be accomplished with relative ease.

I would like to extend my gratitude to my other committee members, Dr. J. P. Chandler and Dr. Nohpill Park. Their knowledgeable inspiration and careful attention to the detail have assured the value and quality of the thesis.

I must express my great appreciation to Mr. Jianhua Ren, who developed the software, SFUDSS 1.0. His creative suggestions, sincere encouragement, and friendship have been invaluable and will remain in my mind.

Finally, the special thanks and apologies should go to my family – my parents and daughter. Over time they have been neglected, even ignored, during my deepest concentration. Their unconditional love and support have been great motivation for my thesis development.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION .....	1
1.1 The Contribution of SFUDSS1.0 to Fertilizer Application.....	1
1.2 Problem Statement .....	2
1.2.1 The Perplexity on Fertilizer Market.....	2
1.2.2 The Complexity in Using Solver.....	4
1.3 Objective of the Study.....	8
II. LITERATURE REVIEW .....	9
2.1 Interprocess Communication.....	9
2.2 Component Object Model .....	10
2.3 ActiveX Controls .....	13
2.4 ADO Data Control and Data-Bound Control.....	14
2.5 Object Link Embedded .....	17
2.6 Visual Basic for Application .....	20
2.7 Excel.....	22
2.8 Linear Programming .....	23
2.9 Excel Solver .....	36
2.10 Database Management System and Relational Database Model .....	37
2.11 Dynamic Link Library.....	39
III. DELOPMENT AND IMPLEMENTATION .....	41
3.1 Overview of the Features of the System .....	41
3.2 Design and Implementation .....	42
3.2.1 Construction of User Graphic Interface .....	42
3.2.2 Design and Implementation of Database .....	46
3.2.3 Algorithm for Selection of Fertilizer Blending.....	53
3.2.4 Access Database from Visual Basic with ADO .....	55
3.2.5 Displaying of Recordset by DataGrid Control.....	56
3.2.6 Realization of OLE Automation .....	57
3.2.7 Implementation of Solver with VBA .....	59
IV. RESULTS .....	63
V. CONCLUSION AND FUTURE WORK.....	69

Chapter	Page
5.1 Summary .....	69
5.2 Conclusions .....	70
5.3 Future Work .....	71
BIBLIOGRAPHY .....	72
APPENDIXES .....	76
Appendix A – Acronyms and Abbreviations .....	76
Appendix B – Glossary .....	78

## LIST OF TABLES

Table	Page
1. Fertilizers with their Ingredients and Contents on the Market.....	3
2. The Results from Different Fertilizer Combinations .....	4

## LIST OF FIGURES

Figure	Page
1. Manually Setting the Spreadsheet .....	6
2. Set Solver Parameter Manually .....	7
3. ADO Object Model .....	15
4. Structure of Optimal Fertilizer Investment System .....	42
5. Relations in the Third Normal Form .....	50
6. E-R Diagram for Fertilizer Database .....	51
7. Crop Tables .....	52
8. Method Table .....	52
9. Fertilizer Table .....	53
10. Mix Table .....	53
11. Starting Form.....	63
12. PreInformation Form.....	64
13. Combination Form .....	66
14. Optimal Results on the Spreadsheet.....	67
15. Optimization of Fertilizer Investment Form .....	68

# Chapter I

## Introduction

The use of fertilizer is needed for all types of long-term crop production in order to achieve yield levels which make the effort of cropping worthwhile [17].

Modern fertilizer practices, first introduced more than a century ago and based on the chemical concept of crop nutrition, have contributed widely to the immense increase in agricultural production and have resulted in better quality food and fodder. As a beneficial side-effect, the fertility of soils has been improved resulting in more stable yield levels as well as in a better (nutrition-induced) resistance to some diseases and climatic stress. Furthermore, the farmer's economic returns have increased due to more effective production [17].

### *1.1 The Contribution of SFUDSS1.0 to Fertilizer Application*

Nutrient deficiency in soil can be corrected simply by applying appropriate fertilizers, but it is neither practicable nor economic to attempt to eliminate deficiency and to maximize crop production through massive applications of fertilizer, since excessive fertilizer into soil would result in environmental risk [39]. For the purpose of optimal fertilization, the Oklahoma State University (OSU) Soil, Water and Forage Analytical Laboratory (SWFAL) has developed a computer-based software, named Decision Support System for Soil Fertilizer 1.0 (SFUDSS 1.0). This software can generate recommendations of fertilizer application, including the amount of the three major nutrients contained in a specified field, such as nitrogen (N), phosphorous (P), and



potassium (K), based on test results and further requirements for reaching an anticipated yield goal.

SFUDSS1.0 is capable of telling farmers how much N, P, and K needed for their planting plan, however it cannot instruct farmers in the amount and sort of fertilizers they should purchase from the most economic perspective.

## **1.2 Problem Statement**

### **1.2.1 The perplexity on fertilizer market**

With the aid of a Decision Support System for fertilizer application, farmers learn fertilizer requirements based on their yield goals. However, the farmers do not know what mixtures of fertilizers are best for them to purchase since the commercial fertilizers on the market contain a variety of ingredients with different prices.

For instance, assume a recommendation that states one field needs at least 32 pounds of nitrogen, 24 pounds of phosphorus, and 42 pounds of potassium. On the market, there are five types of fertilizers available as shown in Table 1.

Fertilizer	Nitrogen	Phosphorous	Potassium	Cost/pound
A	3%	5%	1.4%	\$0.04
B	30%	0%	0%	\$0.15
C	0%	20%	0%	\$0.01
D	15%	10%	7%	\$0.13
E	0%	0%	20%	\$0.14

**Table 1. Fertilizers with their ingredients and contents on the Market.**

These fertilizers can be combined differently for purchase, however the ingredients and the prices resulting from different combinations appear greatly divergent. Table 2 demonstrates this fact.

Fertilizer Mixture	Fertilizer Needed (lbs/a)	Nutrients Contained			Surplus Nutrients			Cost
		N	P	K	N	P	K	
A	3000	90	150	42	58	126	0	\$120
B	107	32.1	24	42	0.1	0	0	\$57.5
C	120							
E	210							
A	88.89	32	24	42	0	0	0	\$47.9
D	195.56							
E	135.33							

**Table 2. The results from different fertilizer combinations.**

If fertilizer A is the only choice, then at least 3000 pounds of this type of fertilizer is necessary to satisfy required 42 pounds of K at the cost of \$ 120.

If a combination of fertilizers occurs among fertilizers B, C and E, then no less than 107 pounds of B, 120 pound of C, and 135 pounds of E are needed to fill the nutrient requirements. This combination produces 32.1 pounds of N, 24 pounds of P, and 42 pounds of K at a price of \$57.50.

If the selection includes fertilizers A, D, and E, then 88.89 pounds of A, 195.56 pound of D, and 135.33 pounds of E are enough to meet the ingredient requirements at a cost of \$ 47.90.

Comparing the three combinations, it is obvious that the first choice is thoroughly undesirable. It leads to not only the waste of 58 pounds of N and 126 pounds of P but also environmental pollution due to excessive Nitrogen and phosphorous applied to the soil. The amount of \$120 charged for this mixture of fertilizers is also not a small investment.

The second choice greatly decreases the waste of N, P and K, and it significantly lowers the cost. However, the price for this combination is unacceptable.

The last choice removes any wasted nutrient ingredients as well as having the least cost. This choice is optimal in the case shown in table 2.

In addition to the consideration of nutrient ingredients in fertilizers, the other factors, such as crop growth phase, fertilizer form, methods of fertilizer placements, fertilizer compatibility, and side-effects of some fertilizers on certain crops, also affect the selection of the fertilizers and must be taken into account.

In general, fertilizers exist in either dry or liquid form. The distinct forms of fertilizers are applied using different methods of fertilizer placement. The crop growth phase determines the utilization of the placement methods.

Finding an ideal combination of the required fertilizer ingredients that are compatible, satisfy the constraints of placement method and crop growth phase, and are provided with the least cost is an optimization problem.

The process to solve the optimization problem is quite difficult and complicated. It involves using a mathematical method to formulate the problem then operating tools to reach an optimal value. At present, Solver built-in Excel is one of the methods widely used to realize optimization.

### 1.2.2 The complexity in using Solver

The built-in Solver in Excel is a powerful tool for computing an optimal solution. The theoretical foundation of Solver is linear programming. The basic concept in linear programming is finding a particular set of variables that satisfy all constraints and maximizes, or minimizes, the value of the objective function. Usually, using Solver to compute an optimal solution takes the following steps:

1) *Formulating an optimization model.*

- a. Select relevant variables and restrictions. Express relationships between variables and restrictions in equation form.

Based on example in table 1, one field needs at least 32 pounds of nitrogen, 24 pounds of phosphorus, and 42 pounds of potassium. The problem can be formulated as follows:

Let  $X_1$  = amount of fertilizer A

$X_2$  = amount of fertilizer B

$X_3$  = amount of fertilizer C

$X_4$  = amount of fertilizer D

$X_5$  = amount of fertilizer E

$$0.03X_1 + 0.30X_2 + 0.15 X_4 \geq 32$$

$$0.05 X_1 + 0.20 X_3 + 0.10 X_4 \geq 24$$

$$0.14 X_1 + 0.07 X_4 + 0.20 X_5 \geq 42$$

$$X_i \geq 0 \text{ (} i = 1, 2, 3, 4, 5 \text{)}$$

- b. Determine an objective function or measure of effectiveness referring to the example in table 1, the objective function is set to:

$$S_{\min} = 0.04 X_1 + 0.15 X_2 + 0.1 X_3 + 0.13 X_4 + 0.14 X_5$$

$$X_i \geq 0 \text{ (} i = 1, 2, 3, 4, 5 \text{)}$$

2) *Setting the optimal model up on a spreadsheet.*

This involves several tasks as follows:

- a. Reserve a cell to hold the value of each decision variable.
- b. Pick a cell to represent the objective function, and enter a formula that calculates the

objective function value in this cell.

- c. Pick other cells and use them to enter the formulas that calculate the left hand sides of the constraints.
- d. The constraints on the right hand side can be entered as numbers in other cells, or entered directly in the Solver Add Constraint dialog box.

Figure 1 and Figure 2, based on the example in table 1, show how a user manually sets the spreadsheet to compute an optimal solution.

- a. Reserve the cells from E<sub>3</sub> to I<sub>3</sub> for as constraints the contents of N contained in the five types of fertilizers. Reserve the cells from E<sub>4</sub> for I<sub>4</sub> as constraints for the contents of P contained in the five types of fertilizers. Reserve the cells from E<sub>5</sub> to I<sub>5</sub> as constraints for the contents of K contained in the five types of fertilizers.
- b. Pick cell C<sub>8</sub> to hold the objective function,  $f = 0.04x_1 + 0.15x_2 + 0.16x_3 + 0.13x_4 + 0.14x_5$ .

	A	B	C	D	E	F	G	H	I
2		NPK need	NPK got		FertA	FertB	FertC	FertD	FertE
3	N	32	0.00		0.03	0.3	0	0.15	0
4	P	24	0.00		0.05	0	0.2	0.1	0
5	K	42	0.00		0.014	0	0	0.07	0.2
6									
7		Amount of fertilizer			0.00	0.00	0.00	0.00	0.00
8		Price of fertilizers			\$0.04	\$0.15	\$0.10	\$0.13	\$0.14
9									
10		Total Cost			\$0.00				
11									
12									

Figure 1. Manually setting the optimization model on the spreadsheet.

- c. Use cells C<sub>3</sub>, C<sub>4</sub>, and C<sub>5</sub> to keep each formula that calculate the value on the left hand side respectively. These formulas take the form of SUMPRODUCT (E<sub>3</sub>:I<sub>3</sub>, E<sub>7</sub>:I<sub>7</sub>), SUMPRODUCT (E<sub>4</sub>:I<sub>4</sub>, E<sub>7</sub>: I<sub>7</sub>), and SUMPRODUCT (E<sub>5</sub>:I<sub>5</sub>, E<sub>7</sub>:I<sub>7</sub>). SUMPRODUCT is a function that multiplies the content of each nutrient in each fertilizer to X<sub>n</sub> (an initial value), respectively, and sums them. For example, SUMPRODUCT computes for nutrient N  $3\%X_1 + 30\%X_2 + 15\%X_4$ .
- d. Enter the constraint on the right hand side in the cells from B<sub>3</sub> to B<sub>5</sub>.

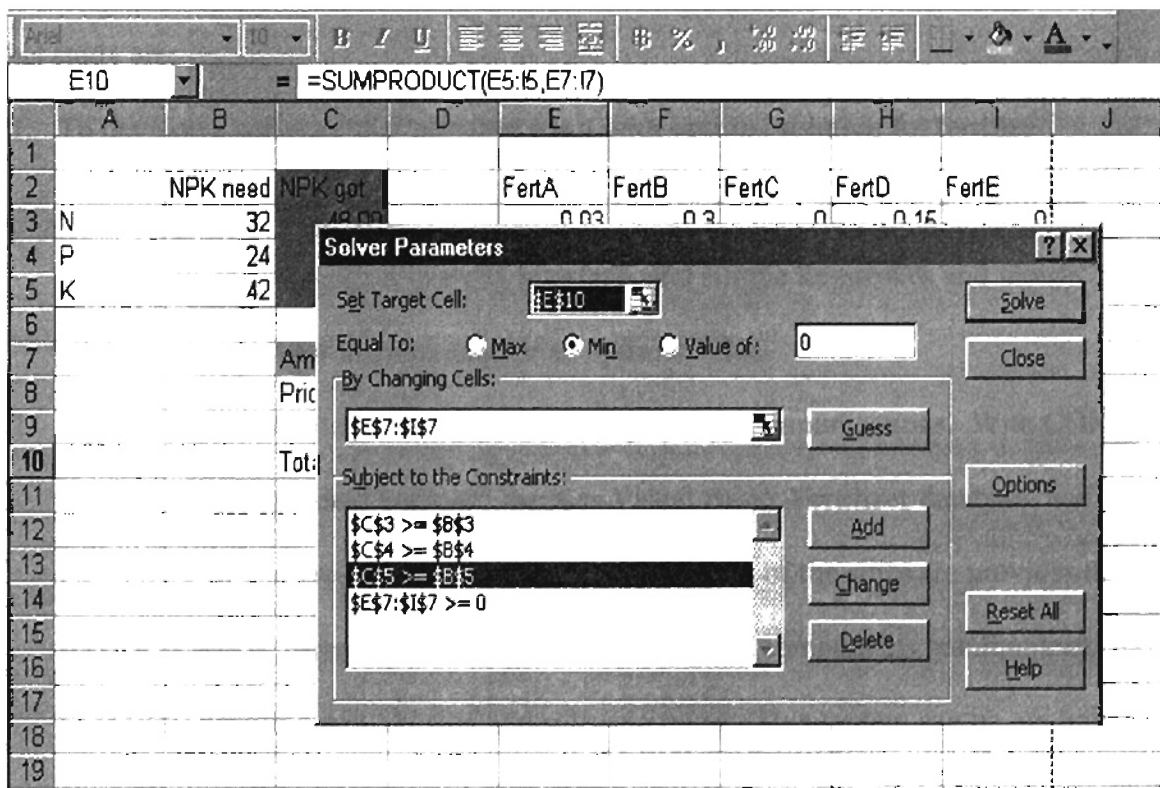


Figure 2. Set Solver parameters manually.

### 3) Activating the Solver to find an optimal solution.

Clicking the Solver parameter dialog box to drive Solver computing optimal solution. Using Excel Solver to compute an optimal solution is a difficult and

complicated process. It requires the users (1) know the fundamentals of linear programming and its solving steps, (2) have the knowledge of advanced Excel, and (3) be capable of manipulating Solver.

Obviously, it is impossible to complete this process for users who lack the knowledge of and experience with linear programming, Excel, and Solver.

### ***1.3 Objective of the Study***

To meet the need of the farmers for an optimal fertilizer investment, the objectives of this study are:

1. To develop a software that provides users with optimal solution for fertilize investment and frees them from difficult tool operation. Users interact with the application only through a friendly interface, and simple keystrokes and mouse clicks are enough for them to acquire an optimal value.
2. To use COM technology to implement interprocess communications. With COM a variety of independent software, such as Visual Basic, Fertilizer database, and Excel used in this project, are integrated, and any part of the software provides its service as one or more COM objects. These software components communicate and cooperate, enabling data sharing and software reuse. This system can be run under any Microsoft Windows System with Office installed.

# Chapter II

## Literature Review

### *2.1 Interprocess Communications*

As computer users become more sophisticated, they demand more power from the applications they use. To meet this demand, developers add more features to the applications, and the applications become larger. Large applications eventually become unmanageable, both from a development standpoint and from a user-interface point of view.

One method a developer can use to manage larger applications is to produce a specialized application that provides a limited number of features, then enable it to communicate and share data with other specialized applications using some form of Interprocess Communications (IPC). It is not necessary that any single application meets all its user's expectations; users can use the communicating and cooperating applications.

Generally, IPC is a capability supported by some operating systems that allows one process to communicate with another process. The processes can be running on the same computer or on different computers connected throughout a network. IPC also enables one application to control another application, and for several applications to share the same data without interfering with one another [24]. IPC is required in all multiprocessing operating systems.

Typically, IPC applications can be categorized as clients or servers. A client is an application or a process that requests a service from some other process. A server is an application or a process that responds to a request. Many applications act as both a client



and a server, depending on the situation. For example, a Visual Basic application might act as a client in requesting a setting of constraints and an objective function from an Excel spreadsheet that acts as a server. The spreadsheet application, in turn, might be a client requesting an optimal computation from Solver.

There are a few IPC mechanisms that are supported by Win32 API (Application Programming Interface), such as Clipboard, COM, Dynamic Data Exchange (DDE), File Mapping, Mailslots, Pipes, Remote Process Call (RPC) and Windows Sockets. Among them COM is used popularly for interprocess communications.

## 2.2 *Component Object Model*

The Component Object Model (COM) is a language-independent software component model that enables interaction between software components and applications running on a Windows platform. COM is the foundation on which Microsoft OLE, ActiveX, and other programming technologies are constructed [38].

COM has following features:

*COM is based on components and objects.* COM is a Component-oriented design that breaks applications into components that can be distributed in separate binary files – either dynamic link libraries (DLL) or executable (EXEs). In this context, components are pre-compiled, interacting pieces of software that can act as building blocks for creating applications [53].

Component-oriented design also makes a given component reusable in different applications that require similar functionality and allows the individual components of an application to be updated easily.

*Interface is the key concept in COM.* The interfaces of a component are the mechanism by which its functionality can be used by another component. Interfaces contain functions. COM defines certain basic interfaces that provide function common to all COM-based technologies [4]. To provide additional functions for a component, simply an additional interface needs to be added from those functions.

*COM components are language-independent.* COM component can be written in any of several different languages with quite dissimilar structures. These components can be called within a single process, in other processes, even on remote machines. For example a user can create COM component in Visual Basic and integrate it into the user's Visual C++ or Visual J++ project. Language-independence makes it possible to build then subsequently use components with distinct languages [41].

*COM components act as both client and server.* The term 'client /server' encapsulates two separate, discrete entities. These entities could simply be two components, two applications or even two computers. The client requires some service, and requests a server to perform this service on its behalf. The server performs the service and returns a result to the client.

In terms of COM components, each component can function both as client and as server. A server is a component that exposes its interfaces and therefore a list of functions that a client may call. The client is a component that uses that interfaces. COM enforces a strict separation of the client and server such that they only know of each other's presence by the existence of their interfaces [20].

*COM objects have object-oriented features.* Obeying the structures of encapsulation, they are accessed using only public methods and properties, so that any data they contain

is hidden from public consumption.

COM is a framework for creating and using components [38]. It offers following advantages:

1. COM provides a mechanism for components to communicate with one another.

COM achieves the communication by the combination of two things.

First, there is the binary standard it imposes. This means that the components not only provide a mechanism to communicate, but also communicate with other COM-enabled components or applications.

Second, there is a set of services provided by the COM runtime library that is required to be present on a computer before communication between the components can occur. These services, which are available to all components, come in the form of a set of Application Programming Interface (API) functions [20].

2. COM components are usually written to be generic, which allows them to be used in multiple ways by different pieces of software [38]. For example, a user can put a graph in a Microsoft Word document, or the user might put it onto a Microsoft Access form. A user can use it in either application and it will work the same way in one as it does in the other.

3. COM promotes the use of component-based development.

Before the advent of component-based development, the predominant technique was procedure-based development. This style is linear in nature, and programs start at the first line of code, and finish either when the last line is executed, or an Exit statement is reached. Applications written in this fashion tend not to be vary adaptive to what was happening around them [4].

In Visual Basic today, it is quite easy to reference (that is, to plug in) a library of functionality in order to use it in users' programs.

Component-based development also allows the programmers to take advantage of using pre-packaged components or tools.

4. COM promotes object-oriented programming.

COM was created with object-oriented programming in mind. This is a methodology that improves thinking about software 'objects' and the way those objects interact with each other, rather than their implementation [41].

As COM has evolved, it has been extended beyond the basic COM service. COM serves as the basis for other technologies, such as Automation and ActiveX controls.

### **2.3 ActiveX Controls**

ActiveX controls are software components that are used to realize software reuse and increase productivity. These controls use COM technologies to provide interoperability with other types of COM components and services [33].

ActiveX controls are visual controls that can be plugged into an ActiveX control container application. They are not complete applications in themselves, but can be thought of as prefabricated OLE controls that are reusable in various applications [24]. ActiveX controls have a visible user interface, and rely on these predefined interfaces to negotiate I/O and display issues with their host containers [9].

An ActiveX control encapsulates, or contains, three different parts --- properties, methods, and events --- that users will modify, call, and define to take advantage of the control's functionality in their programs:

- Properties define attributes of a control, such as the way a control looks on the form or the initial state of the control when a users runs the program.
- Methods are functions that perform a specific action on or with an object.
- Events are notifications generated by an ActiveX control in response to some particular occurrence in the program, such as a mouse click on a user interface control or a completed acquisition.

ActiveX controls make use of automation to expose their properties, methods, and events. Features of ActiveX controls include the abilities to fire events, bind to data sources, and support licensing.

#### ***2.4 ADO Data Control and Data-Bound Control***

*ADO Data Control* is a form of ActiveX controls. It is based on the ActiveX Data Objects (ADO). ADO is a high-level interface to all kinds of data and is used to provide consistent access to the data [25].

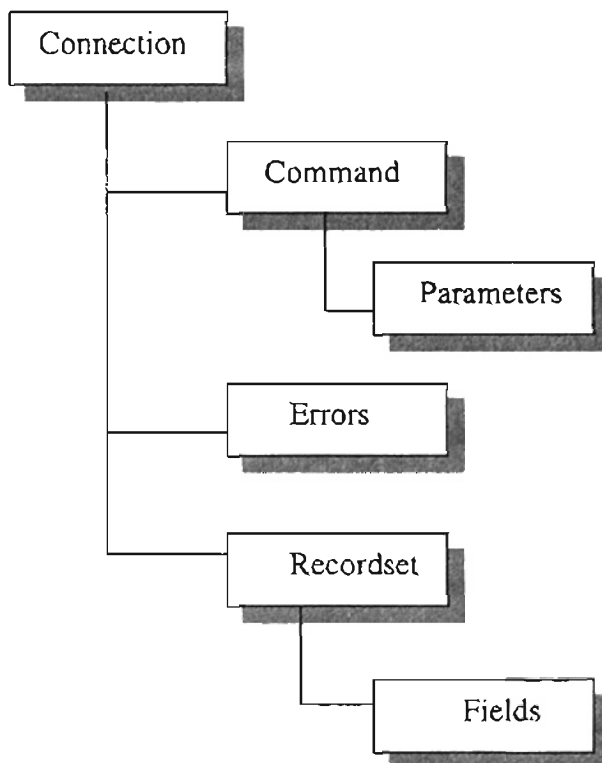
ADO sits on the top of database. It hides the peculiarities of each database, and gives developers a simple conceptual view of the underlying database [36].

ADO enables client applications to access and manipulate data through any OLE DB provider, and it contains object for connecting to a data source, adding, updating, or deleting data. A client application performs the following:

1. Establishes a connection to the database
2. Executes commands against the database
3. Retrieves information from the database

ADO object mode, shown in Figure 3, corresponds to these operations.

- ◆ *Connection object* builds a connection to the database.
- ◆ *Command object* points to SQL strings, stored procedures, or action queries that a user can execute.
- ◆ *Recordset object* creates a set of records from a query (User can move forward and backward through ADO Recordsets. ADO Recordset is also called cursors) and it can be used to view the contents of a table or results from executing an SQL statement.
- ◆ *Parameter object* represents a parameter of a command. A user explicitly can generate parameter objects then add them to the parameter collection to avoid the unnecessary and expensive task of going to the system catalog to populate the parameter binding information automatically.
- ◆ *Field object* displays a column in a Recordset.



**Figure 3. The top-level object in ADO is the Connection object, which contains Command and Recordset objects and the Errors collection.**

- ◆ *Error object* deals with an error returned from a data source. This object is optional because it is only needed when data sources can return multiple errors for a single method call.

*ADO Data Control* represents the ActiveX Data Objects, and it accesses databases through the ADO objects. ADO Data Control is placed on a Form and functions as a user application's visual gateway to a database. A user can set it up to "see" any table or query in a database with point-and-click operations.

ADO Data Control exposes the basic properties of the Connection object, which allow a user to connect to a database, and the basic properties of the Command object, which allow a user to retrieve the desired rows. To access these rows, use the Recordset object. This is a property of the ADO Data Control, which in turn exposes the same properties as the ADO Recordset object.

*Data-Bound Control* is another form of ActiveX control. It comes with Visual Basic and can bind to multiple rows in a table. These controls are also called Data-aware controls and must be used with a data source. Working with ADO data control, data-bound controls help a user to view and update a specified field in the database.

Data-bound controls, described below, can only be used in conjunction with a database:

Datalist control This is similar to the ListBox control, but its items are the values of a column in the database (or a query).

A ListBox control presents lists of choices for user to select. The ListBox control

occupies a user-specified amount of space on the Form and is populated with a list of items; the user can select one or more with the mouse.

DataCombo control This is similar to the ComboBox control, but its items are the value of a column in the database (or a query).

A ComboBox control contains multiple items but occupies less space on the screen. The ComboBox control is an expandable ListBox control: the user can expand it to make a selection and retract it after the selection is made.

DataGrid control It maps an entire table (or query) on a grid. DataGrid control calls direct editing of its cells, as well as the addition of new rows and the deletion of existing ones.

The DataList and DataCombo controls are frequently used as lookup mechanisms. They display the values of a specific column, yet they can be bound to another Recordset and synchronize their data with the second Recordset.

## ***2.5 Object Link Embedded***

In the past, software development tools were uniform throughout, allowing little variation and component exchange with other tools. Programming environments were thought of as islands, somewhat isolated from other applications. Programmers used structured programming techniques to analyze programming problems in terms of procedures and then implement those procedures [28].

With the introduction of Object Link Embedded (OLE), software development has been significantly transformed from procedural to object-oriented programming. The developers no longer have to work with prepackaged tools. They can create self-



contained models or objects that greatly simplify programming, especially when it comes to building large applications [28]. With OLE developers can use both Visual Basic objects and objects exposed by other applications to build new applications. They will be an integral part of the operating system and its applications [28].

OLE is unified environment of object-based service. Its essence is allowing applications to supply their objects to external development tools, macro languages and other applications that support OLE.

At core of OLE is the Component Object Model ( COM ). COM is a standard software architecture based on interfaces that is designed to separate code into self-contained objects or components. Each component exposes a set of interfaces through which all communication to the component is handled.

To realize OLE, following applications and techniques are necessary:

*OLE Object* – an item that is exposed, or made available, by and OLE server application. OLE server application exposes different type (or classes) of objects. OLE objects are used in container applications.

*Server Application* – an application that exposes the objects a Visual Basic application contacts. When an application must edit a document created by a server application, it contacts the server application used to edit the document.

*Container Application* – an application that contains the OLE objects. Objects can be linked or embedded. A container itself is also an object. The container is also referred to as a *client* because it uses the services of OLE servers to obtain the objects.

*Object Embedding* – a technique with which we can insert an object from one application (the server application) into another application (the container application).

The inserted object is a copy of the original and can be manipulated and stored separately and apart from the original object.

*Object Linking* – a technique similar to embedding, except that the embedded data are also *linked* to the document from which they come. Changes to the object in the server application are reflected automatically in the container application. Linking doesn't store the object, it makes a reference to the object exposed by the server application. Each time opening the document that contains the linked object, the container application contacts the server application, which actually opens the most up-to-date version of the linked object. Linked objects are not copies. They are the originals, viewed from within different containers.

*In-Place Editing* – also known as in-place activation. The functionality of the server application is incorporated into the container application, thus enabling us to edit the object using the menus and tools of the server application.

*OLE Drag- und-Drop* – a method allows users to pick up objects that have been exposed in a server application and place or drop them into users' container application.

*OLE Automation* – a method that allows manipulation of objects exposed by another application from within Visual Basic applications. It is also a standard that defines how code is shared between applications and how applications can be controlled from within other applications.

Generally, using OLE Automation, users can:

- Create applications and programming tools that exposed objects.
- Create and manipulate objects exposed in one application from another application.
- Create tools that access and manipulate objects. These tools can include embedded

macro languages, external programming tools, object browsers, and compilers.

The objects an application or programming tool exposes are called OLE Automation objects. Applications and programming tools that access the objects are called OLE Automation controllers. These controllers can be created using Microsoft Visual Basic, Microsoft Visual C++, Microsoft Excel, Microsoft Project, and other applications and programming languages that support OLE.

## ***2.6 Visual Basic for Applications***

When Microsoft developed OLE Automation, the basic idea was simple: create a common language and programming environment for a number of applications so that users could customize applications and add capabilities to suit their own environments. The result was the language Visual Basic for Applications [30].

*Visual Basic for Application (VBA)* is powerful language and development environment built into the Microsoft Office family of applications, which provide developer with professional quality development tools for building custom solutions [19].

VBA comprises a VBA engine and an integrated development environment (IDE) with a full-featured editor, debugger, and OLE browser. It supplies basic control structures, mathematical and string functions, and variable manipulation capabilities that enables developers to learn a single language and development environment which can then be used across multiple applications.

VBA is an object-oriented environment that provides a large collection of objects, each with its own sets of properties and methods. This environment makes automation and application re-use possible.

VBA programming environment can be incorporated into applications that support Automation to make them programmable. The suite of Microsoft Office applications incorporate the VBA programming environment and are written to support Automation interfaces [26]. In addition, many other software components, such as ActiveX controls and DLLs (Dynamic Link Libraries), also expose their functionality to VBA programmers through Automation interfaces.

Using the objects, properties, and methods exposed through Automation interfaces, one can use VBA code running in modules associated with the currently open document, template, database, Microsoft FrontPage-based web, or add-in to automate that application [26]. VBA and Automation make it possible to record simple macros to automate keystrokes and mouse actions, and also to create sophisticated integrated solutions, such as document management, accounting, and database applications [19].

To produce even more powerful integrated applications, one can use VBA code running in one application to create and work with objects from another installed application or component. For example, if a user is developing a solution in Access and wants to use mathematical or other functions available only in Excel, the user can use VBA to create an instance of Excel and use its features from code running in Access.

Generally, Automation can be thought of as a nervous system that makes programmatic communication and feedback between applications and components possible, and “glues” that lets you integrate features from Office applications and other software components into a custom solution [35].

VBA’s support for Automation provides Office developers with incredible flexibility and power. By taking advantage of Automation, a user can use the features exposed

through the object models of the entire Office suite of applications (as well as any third-party applications and components that support Automation interfaces) as a set of business-application building blocks [12]. By taking advantage of the pre-built components exposed through Automation, a user doesn't need to develop own custom components and procedures each time the user wants to get something done. It greatly saves the time for developing the user's solution.

## ***2.7 Excel***

Microsoft Excel was developed as a business application for accounting and financial calculations. Excel, however, is also powerful to perform many of calculations that scientists and engineers use. Excel has sufficient numeric precision to represent most of the values in calculations of interest to those in technical fields. There are also facilities for creating user-defined functions and automating often-repeated task/calculations with macros [18].

Excel is able to create and perform "what-if" analysis of complex, interrelated columnar reports, also called spreadsheets or worksheets. Spreadsheets are made of cells arranged in rows and columns that can hold labels, numbers, formula and data. Excel gains its popularity due to this feature that makes information easy to present in meaningful way.

Excel is the leading spreadsheet program, providing features for:

- handling numbers and text
- adding formulas and functions to automatically carry out calculations
- graphing data in the spreadsheet

- generating statistical analysis of the data

The spreadsheet has a reference to Solver Add-in, so that all the Solver functions will be available to Excel users. The spreadsheet also is able to function as an interface that holds inputs and outputs for Excel Solver.

In 1993, Microsoft incorporated customized version of Visual Basic for Applications into Excel, therefore it becomes possible to operate Solver from other applications.

## 2.8 Linear Programming

Linear programming (LP) is used as a standard tool to allocate a finite set of resources in an optimal way. Linear programming is applied to a specific class of mathematical problems, in which a linear function is either maximized or minimized subject to given linear constraints. In another words, the basic problem in linear programming is to find the particular set of variables that satisfies all constraints and maximizes or minimize the value of the objective function. Any problem that fits the two basic assumptions, linearity and certainty, can be solved by linear programming [45].

A linear program can be expressed as follows (the so-called Standard Form):

$$\text{Minimize } f = c_1x_1 + c_2x_2 + \dots + c_nx_n \text{ (objective function)}$$

Subject to:

$$1^{\text{st}} \text{ constraint: } a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = b_1$$

$$2^{\text{nd}} \text{ constraint: } a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = b_2$$

...

...

$$m^{\text{th}} \text{ constraint: } a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n}x_n = b_m$$

positivity:  $x_1 \geq 0, x_2 \geq 0 \dots x_n \geq 0$

where  $f$  is the function to be minimized, the  $x$ 's are variables, the value of which should be determined in such a way that  $f$  is minimized and the  $c$ 's are given fixed coefficients.

The importance of linear programming derives in part from its many applications and in part from the existence of good general-purpose techniques for finding optimal solutions. These techniques take as input only an LP problem in the above standard form, then determine a solution without reference to any information concerning the LP's origins or special structure. They are fast and reliable over a substantial range of problem sizes and applications [46].

The simplex method has been the standard technique for solving a linear program since the 1940's. It was introduced by Dantzig. The simplex method passes from vertex to vertex on the boundary of the feasible polyhedron, repeatedly decreasing the objective function until either an optimal solution is found, or it is established that no solution exists. In practice, the method is highly efficient, typically requiring a number of steps that is just a small multiple of the number of variables [32]. Linear programs in thousands or even millions of variables are routinely solved using the simplex method on modern computers. Efficient, highly sophisticated implementations are available in the form of computer software packages [45], such as Excel Solver add-in.

The following definitions are involved in the simplex method:

*Objective Function:* the function that is either being minimized or maximized.

*Optimal Solution:* a vector,  $x$ , which is both feasible (satisfying the constraints) and optimal (obtaining the largest or the smallest objective value).

*Constraints:* a set of equalities and inequalities that the feasible solution must satisfy.

*Feasible Solution:* a solution vector,  $x$ , which satisfies the constraints.

*Canonical System:* an equation system in which a variable that has a unit coefficient in one equation and zeros in other equations.

*Slack Variable:* a variable added to the equations to eliminate less-than constraints.

*Surplus Variable:* a variable added to the equations to eliminate greater-than constraints.

*Artificial Variable:* a variable used to get a canonical system with a basic feasible solution when none is available in the equation.

*Basis:* a set of basic variable.

*Basic Variable:* a variable that appears with a unit coefficient in a equation and zeros in all other equations. Basic variable is in the basic solution.

*Nonbasic Variables:* a variable that is not in the basic solution and its value is zero.

*Pivot Operation:* a sequence of elementary operations that reduces a given system to an equivalent system in which a specified variable has a unit coefficient in one equation and zeros elsewhere.

*Basic Solution:* the solution obtained from a canonical system by setting the nonbasic variables to zero and solving for the basic variables.

*Basic Feasible Solution:* a basic solution in which the values of the basic variables are nonnegative.

*Relative Profit:* the net change in the value of objective function per unit increase ( for maximization) or decrease (for minimization) in a nonbasic variable.

$$\text{Relative Profit (on } x_n) = \frac{\text{New value of objective function} - \text{Old value of objective function}}{x_n}$$



*Minimum Ratio Rule:* a rule used to determine the variable to leave the basic set, involving the calculation of ratios and selection of the minimum ratio.

*Pivot Row:* the row that contains the minimum ratio.

*Solution Set:* the collection of all possible solutions to the system.

The principles adopted by simplex method include [37]:

1. Start with an initial basic feasible solution in canonical form.
2. Improve the initial solution if possible by finding another basic feasible solution with a better objective function value. At this step the simplex method implicitly eliminates from consideration all those basic feasible solutions whose objective function values are worse than the present one.
3. Continue to find better basic feasible solutions improving the objective function values. When a particular basic feasible solution cannot be improved further, it becomes an optimal solution and the simplex method terminates.

The steps of the simplex method for either maximization or minimization problem are as follows:

**Step I.** Frame the problem.

- a. Select relevant variables and restrictions. Express relationships among all variables and restrictions in standard (equation) form.
- b. Determine an objective function.

**Step II.** Start with an initial basic feasible solution in canonical form.

**Step III.** Check whether the current basic feasible solution is optimal. At this solution, the relative profits of all the nonbasic variables are computed. If these coefficients are

negative (for maximization), or positive (for minimization), or zero, the current solution is optimal. Otherwise, go to Step IV.

**Step IV.** Select a nonbasic variable to be the new basic variable in the solution. A general rule is to select the nonbasic variable with the largest relative profit (most positive value for maximization) or smallest relative profit (most negative value for minimization) so that it may give an larger increase or decrease in the value of the objective function.

**Step V.** Determine the basic variables to be replaced by the nonbasic variable.

Examining each of the constraints to determine how far the nonbasic variable can be increased or decreased. For these constraints in which the nonbasic variable has a positive coefficient, the limit is given by the ratio of the right-hand side constant to that positive coefficient. For the other constraints the limit is set to  $+\infty$ . The constraint with the lowest limit is determined, and the basic variable in that constraint is replaced by the nonbasic variable.

**Step VI.** Find the new canonical system and the basic feasible solution using a pivot operation. Return to Step III.

The various steps of the simplex method can be carried out by using a tableau form to represent the constraints and the objective function. The use of the tableau form has made the simplex method more efficient and convenient for computer implementation.

The following problem is solved by the simplex method using a tableau form.

The problem states that each fertilizer contains different content of N, P and K. In the fertilizer mixture, the total content of N must greater or equal to 32 pounds, the total content of P should be greater or equal to 24 pound, and the total content of K is greater

or equals to 42 pounds. This problem requires the purchase of the fertilizer mixture with the least cost.

The problem is defined as follows:

$n$  = the type of a fertilizer ( $n = 1$  for fertilizer 1,  $n = 2$  for fertilizer 2, and  $n = 3$  for fertilizer 3)

$m$  = the nutrient ingredient contained in a fertilizer ( $m = 1$  for N,  $m = 2$  for P, and  $m = 3$  for K)

$a_{mn}$  = the total content of the  $m_{th}$  ingredient in the fertilizer  $n_{th}$

$b_m$  = the total minimum number of pounds of the  $m_{th}$  ingredient required

$c_n$  = the cost per pound of the  $n_{th}$  fertilizer

$x_n$  = the quantum of the  $n_{th}$  fertilizer that should be purchased

This linear-programming problem involves minimizing the cost function

$$c_1x_1 + c_2x_2 + c_3x_3$$

Subject to the conditions

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_3 \geq 0$$

and

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \geq 32$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \geq 24$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \geq 42$$

Assume Fertilizer 1 contains:

$$N = 0.03 \text{ pound, } P = 0.05 \text{ pound, } K = 0.14 \text{ pound, and cost} = \$0.2$$

Fertilizer 2 contains:

N = 0.3 pound, P = 0.2 pound, K = 0.07 pound, and cost = \$0.8

Fertilizer 3 contains:

N = 0.15 pound, P = 0.1 pound, K = 0.2 pound, and cost = \$0.65

Based on the information above:

1. Frame problem.

$$\text{Minimize: } S = 0.2X_1 + 0.8X_2 + 0.65X_3$$

$$\text{Subject to: } 0.03X_1 + 0.3X_2 + 0.15X_3 \geq 32$$

$$0.05X_1 + 0.2X_2 + 0.1X_3 \geq 24$$

$$0.14X_1 + 0.07X_2 + 0.2X_3 \geq 42$$

Where  $X_i \geq 0$  ( $i = 1,2,3$ ).

2. Convert the problem to a standard form

Add slack variables  $X_4$ ,  $X_5$ , and  $X_6$  to equations to eliminate greater-than constraints.

$$\text{Minimize: } S = 0.2 X_1 + 0.8 X_2 + 0.65 X_3$$

$$\text{Subject to: } 0.03 X_1 + 0.3 X_2 + 0.15 X_3 - X_4 = 32$$

$$0.05 X_1 + 0.2 X_2 + 0.1 X_3 - X_5 = 24$$

$$0.14 X_1 + 0.07 X_2 + 0.2 X_3 - X_6 = 42$$

Where  $X_i \geq 0$  ( $i = 1,2,3,4,5,6$ ).

3. Add the artificial variables  $X_7$ ,  $X_8$ , and  $X_9$  to make a canonical form.

$$\text{Minimize: } S = 0.2 X_1 + 0.8 X_2 + 0.65 X_3 + 1000X_7 + 1000X_8 + 1000X_9$$

$$\text{Subject to: } 0.03 X_1 + 0.3 X_2 + 0.15 X_3 - X_4 + X_7 = 32$$

$$0.05 X_1 + 0.2 X_2 + 0.1 X_3 - X_5 + X_8 = 24$$

$$0.14 X_1 + 0.07 X_2 + 0.2 X_3 - X_6 + X_9 = 42$$

Where  $X_i \geq 0 (i = 1, 2, 3, 4, 5, 6, 7, 8, 9)$ .

In objective function, each of artificial variables is assigned a very large value (\$1000). The simplex method, while trying to improve the objective function, will find the artificial variables uneconomical to maintain as basic variables with negative values. Hence they are quickly replaced in the basis by the real variables with smaller values.

#### 4. Start with an initial basic feasible solution

Tableau 1 illustrates the initial basic feasible solution of above problem.

**Tableau 1**  
(INITIAL SOLUTION)

$C_B$	Basis	$C_j$									Constant	Ratio
		0.2	0.8	0.65	0	0	0	1000	1000	1000		
		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$		
1000	$X_7$	0.03	0.3	0.15	-1	0	0	1	0	0	32	106
1000	$X_8$	0.05	0.2	0.1	0	-1	0	0	1	0	24	120
1000	$X_9$	0.14	0.07	0.2	0	0	-1	0	0	1	42	600
$\bar{C}$	Row	-219.8	-569.2	-449.3	1000	1000	1000	0	0	0	Z = \$98000	

In Tableau 1, *Basis* refers to the basis variables in the current basic feasible solution. The values of the basic variables are given under the column *Constants*. The symbol  $C_j$  denotes the cost of the variable  $X_n$  in the objective function, while  $C_B$  denotes the cost of the basic variables.  $Z$  is the value of objective function that is the total minimum cost for the fertilizers' mixture.

From the above table, the basic feasible solution is written as  $X_7 = 32$ ,  $X_8 = 24$ ,  $X_9 = 42$ , and  $X_1 = X_2 = X_3 = X_4 = X_5 = X_6 = 0$ . The value of the objective function is given by the inner product of the vectors  $C_B$  and *constants* as follows

$$Z = (1000 \ 1000 \ 1000) \begin{pmatrix} 32 \\ 24 \\ 42 \end{pmatrix} = 98000$$

5. Check feasible solution

In order to check if the above basic feasible solution is optimal, the relative profits of all the nonbasic variable is calculated using a formula known as *inner product rule*.

The relative profit coefficient of the variable  $X_n$ , denoted by  $C_j$  is given by:

$$\bar{C}_j = C_j - \left[ \begin{array}{l} \text{inner product of } C_B, \text{ and the column} \\ \text{corresponding to } X_j, \text{ in the canonical system} \end{array} \right]$$

For example

$$\bar{C}_{x_1} = 0.2 - (1000 * 0.03 + 1000 * 0.05 + 1000 * 0.14) = - 219.8$$

$$\bar{C}_{x_2} = 0.8 - (1000 * 0.3 + 1000 * 0.2 + 1000 * 0.07) = - 569.2$$

$$\bar{C}_{x_7} = 1000 - (1000 * 1 + 1000 * 0 + 1000 * 0) = 0$$

Since there are some negative values in the  $\bar{C}$  row, the current basic feasible solution is not optimal. The nonbasic variable  $X_2$  with the most negative value gives the largest per unit decrease in  $Z$  and hence it is chosen as the new variable to enter the basis.

6. Determine the leaving variable from the basis and build new tableau

In order to decide which basic variable is going to be replaced, minimum ratio rule is applied to calculate the limits for each constraint.

Row Number	Basic Variable	Upper Limit on $X_2$
1	$X_7$	$32/0.3 = 106.66$
2	$X_8$	$24/0.2 = 120$
3	$X_9$	$42/0.07 = 600$

The minimum ratio appears in the first row, which is called the *pivot row*. Thus when the nonbasic variable  $X_2$  increases to its maximum of 106.66 units, the basic variable in the pivot row ( $X_7$ ) reduces to zero. The new basis contains  $X_2$ ,  $X_8$ , and  $X_9$  as basic variables. The new canonical system is obtained by performing a pivot operation:

1. Divide the pivot row by 0.3 to make the coefficient of  $X_2$  unity.
2. Multiply the pivot row by  $(-0.2/0.3)$  and add it to row 2 to eliminate  $X_8$ .
3. Multiply the pivot row by  $(-0.2/0.07)$  and add it to row 3 to eliminate  $X_9$ .

From Tableau 2 the new basic feasible solution is given by  $X_1 = 0$ ,  $X_2 = 106.7$ ,  $X_3 = 0$ ,  $X_4 = 0$ ,  $X_5 = 0$ ,  $X_6 = 0$ ,  $X_7 = 0$ ,  $X_8 = 2.7$ ,  $X_9 = 34.5$ , and  $Z = \$37285.3$ . In order to check whether this solution is optimal, the new relative profit coefficients have to be calculated. This can be done by applying the inner product rule. On the other hand, it is also possible to calculate the  $\bar{C}$ -row coefficients through the pivot operation. Since  $X_2$  are the new basic variable, its relative profit coefficient in Tableau 2 should become zero. This is done by either inner product operation or pivot operation.

**Tableau 2**

$C_B$	Basis $C_j$	0.2	0.8	0.65	0	0	0	1000	1000	1000	Constant	Ratio
		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$		
0.8	$X_2$	0.1	1	0.5	-3.3	0	0	3.3	0	0	106.7	-32
1000	$X_8$	0	0	0.0	0.7	-1	0	-0.7	1	0	2.7	4
1000	$X_9$	0.1	0	0.2	0.2	0	-1.0	-0.2	0	1	34.5	148
$\bar{C}$ Row		-162.9	0	-164.8	-897.3	1000	1000	1897.3	0	0	$Z = \$37285.3$	

Since  $\bar{C}$  is negative, Tableau 2 is not optimal. An improvement in the objective function may be obtained by making  $X_4$  a basic variable. The minimum ratio rule is used to determine  $X_8$  as a former basic variable leaves the basis. At this stage,  $Z$  is 37285.3.

**Tableau 3**

$C_B$	Basis $C_j$	0.2	0.8	0.65	0	0	0	1000	1000	1000	Constant	Ratio
		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$		
0.8	$X_2$	0.25	1	0.5	0	-5	0	0	5	0	120	-24
0	$X_4$	0.05	0	0	1	-1.5	0	-1	1.5	0	4	-2.7
1000	$X_9$	0.12	0	0.17	0	<u>0.35</u>	-1	0	-0.35	1	33.7	96
$\bar{C}$ Row		-122.5	0	-164.8	0	-346	1000	1000	1346	0	Z = \$33696	

By this step, the value of objective function decreases to 33696. However, Tableau 3 is not optimal, since  $\bar{C}$  is negative. The further improvement is achieved by having  $X_5$  as a basic variable. The minimum ratio rule is applied to determine  $X_9$ , the former basic variable, should leave the basis. Tableau 4 represents a new system with  $X_2$ ,  $X_4$ , and  $X_5$  as new basic variables.

**Tableau 4**

$C_B$	Basis $C_j$	0.2	0.8	0.65	0	0	0	1000	1000	1000	Constant	Ratio
		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$		
0.8	$X_2$	2	1	2.86	0	0	-14.29	0	0	14.29	600	210
0	$X_4$	0.57	0	0.7	1	0	-4.29	-1	0	4.29	148	209.3
0	$X_5$	0.35	0	<u>0.47</u>	0	1	-2.86	0	-1	2.86	96	203.6
$\bar{C}$ Row		-1.4	0	-1.64	0	0	11.4	1000	1000	989	Z = \$480	

The value in  $\bar{C}$  row is negative, therefore Tableau 4 is not optimal.  $X_3$  is selected to substitutes for  $X_5$  as a basic variable and the value for  $Z$  reduces to 480.



**Tableau 5**

$C_B$	Basis	$C_j$									Constant	Ratio
		0.2	0.8	0.65	0	0	0	1000	1000	1000		
		$X_1$	$X_2$	$X_2$	$X_1$	$X_2$	$X_1$	$X_4$	$X_5$	$X_6$		
0.8	$X_2$	-0.12	1	0	0	-6.06	3.03	0	6.60	-3.03	18.18	150
0	$X_4$	<u>0.05</u>	0	0	1	-1.5	0	-1	1.5	0	4	88.89
0.65	$X_1$	0.74	0	1	0	2.12	-6.06	0	-2.12	6.06	203.6	274.3
$\bar{C}$ Row		-0.19	0	0	0	3.47	1.51	1000	997	998	Z = \$146.9	

Tableau 5 contains negative value in  $\bar{C}$  row, therefore Tableau 5 is not optimal. The nonbasic variable  $X_1$  replaces  $X_4$  as new basic variable and the value in Z decreases to 146.9.

**Tableau 6**

$C_B$	Basis	$C_j$									Constant	Ratio	
		0.2	0.8	0.65	0	0	0	1000	1000	1000			
		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$X_9$			
0.8	$X_2$	0	1	0	2.69	-10.10	3.03	-2.69	10	10	-3.03	28.9	-2.87
0.2	$X_1$	1	0	0	22.22	-33.33	0	-22.22	33.33	0	88.88	-2.67	
0.65	$X_3$	0	0	1	-16.49	<u>26.87</u>	-6.06	16.5	-26.87	6.06	137.64	5.13	
$\bar{C}$ Row		0	0	0	4.12	-2.71	1.51	995.9	1003	999	Z = \$130.4		

Tableau 6 is not optimal since a negative value is still present in  $\bar{C}$  row. To improve the objective function,  $X_5$  is selected as a basic variable. The calculation of minimum ratio determines  $X_3$  as a victim leaves the basis. The value in objective function is improved to 130.4.

7. Reach the optimal solution

In Tableau 7, all the coefficients of the  $\bar{C}$  row are nonnegative. This implies that no further improvement in the objective function is possible. Hence, the current basic feasible solution in the objective function  $A = 259.6$ ,  $B = 80.7$ ,  $C = 0$ ,  $X_1 = 0$ ,  $X_2 = 5.12$ ,

$X_3 = 0$ ,  $X_4 = 0$ ,  $X_5 = 0$ , and  $X_6 = 0$  is an optimal solution, and Z with 116.49 is the optimal value for the linear program.

**Tableau 7**

$C_B$	Basis	$C_j$									Constant	Ratio
		0.2	0.8	0.65	0	0	0	1000	1000	1000		
		$X_2$	$X_2$	$X_2$	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$		
0.8	$X_2$	0	1	0.38	-3.51	0	0.75	3.51	0	-0.75	80.7	
0.2	$X_1$	1	0	1.75	-1	0	-7.52	-1.75	0	7.52	259.6	
0	$X_5$	0	0	-0.61	-0.6	1	-0.23	0.61	-1	0.23	5.12	
$\bar{C}$ Row		0	0	0.10	2.46	0	0.9	997.5	1000	1000	Z = \$116.49	

In summary, the computational steps of the simplex method in tableau form for a maximization or minimization problem are as follows [37]:

- Step I. Express the problem in standard form.
- Step II. Start with an initial basic feasible solution in canonical form and set up the initial tableau.
- Step III. Use the inner product rule to find the relative profit coefficients ( $\bar{C}$  row).
- Step IV. If all the  $\bar{C}$  coefficient are nonpositive (for maximization) or nonnegative (for minimization), the current basic feasible solution is optimal. Otherwise, select the nonbasic variable with the most positive (for maximization) or most negative (for minimization) value to enter the basis.
- Step V. Apply the minimum ratio rule to determine the basic variable to leave the basis.
- Step VI. Perform the pivot operation to get the new tableau and the basic feasible solution.
- Step VII. Compute the relative profit coefficients by using the pivot operation or the inner product rule. Return to Step IV.

## 2.9 Excel Solver

Microsoft Excel Solver is a Microsoft Excel add-in for optimization. An add-in is a software program that extends the capabilities of larger programs. For example, there are many Excel add-ins designed to complement the basic functionality offered by Excel.

Solver add-in consists of following files: Solver.xla, Solver32.dll, and Solvsamp.xls. These files calculate solutions to what-if scenarios based on adjustable cells, constraint cells, or cells that must be either maximized or minimized.

Solver add-in determines the optimum value for a formula in a particular cell, called a target cell, on a Microsoft Excel worksheet. Solver adjusts the values of other cells that are related to the target cell using an equation. After the user constructs an equation and defines a set of parameters or constraints for the variables in the equation, Solver uses the following elements to “solve” an equation:

*Target cell* The target cell is the objective. It is the cell in the worksheet model that is minimized, maximized, or set to a certain value.

*Changing cells* Changing cells are the decision variables. These cells affect the value of the target cell. These cells are changed by Solver to find the optimum solution for the target cell.

*Constraints* Constraints are restriction on the contents of cells. For example, one cell in a worksheet model may be restricted to integer values, while another cell may be restricted to being less than a given value.

To find optimal solution, Solver proceeds by first finding a feasible solution, and then seeking to improve upon it, changing the decision variable to move from one

feasible solution to another until the objective function has reached its goal – optimal value. The optimal value is displayed in the worksheet.

Solver offers many functions, however, the following three functions are fundamental to create and solve a model: SolverOK, SolverSolve, and SolverFinish. Solver strongly relies on the spreadsheet to hold the input and output for its calculation. A user can automate the creation and manipulation of Solver models by using a Visual Basic for Applications macro.

## **2.10 Database Management System and Relational Database Model**

A Database is a shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization [9]. Database Management System (DBMS) is a software system that enables users to define, create, and maintain the database and provides controlled access to this database [9].

Typically, a DBMS provides the following facilities:

- It allows users to define the database, usually through a Data Definition Language (DDL). The DDL allows users to specify the data types and structures, and the constraints on the data stored in the database.
- It allows users to insert, update, delete and retrieve data from the database, usually through a Data Manipulation Language (DML). Having a central repository for all data and data descriptions allows the DML to provide a general enquiry facility to this data, called a query language [9]. The most popular query language is Structure Query Language (SQL), which is now both the standard and *de facto* language for relational DBMSs.

- It provides controlled access to the database. For example, it may provide a security system, an integrity system, a concurrency control system, and a recover control system, etc.

*Relational Database Model* is based on the concept of mathematical relations. In the relational model, data and relationships are represented as tables, each of which consists of series of row/column intersections with a unique name. Tables (or relations) are related to each other by sharing a common entity characteristic. The relation type is shown in a relational schema. A table yields complete data and structural independence because it is a purely logical structure [49].

The relational database model has following advantages:

- (1) Because the relational database model achieves both data independence and structural independence, it becomes much easier to design the database and to manage its contents.
- (2) Less programming effort is required since the relational database model has a powerful query language, SQL, which makes ad hoc queries possible.
- (3) The redundancy in the stored data (information) is reduced to minimum (by normalization theorems) to save storage space and to simplify updating(insertion, deletion, and updating) processes. The simplified updating process should result in easy maintenance of data integrity [27].
- (4) The dependencies between attributes are preserved [27].
- (5) The loss-less join properties are maintained during decomposition, such that the data (information) can be easily retrieved without distortion [27].

Microsoft Access is a typical relational data model. It is used to manage data in three important ways:

- ◆ Reducing redundancy.
- ◆ Facilitating data sharing.
- ◆ Keeping data accurate.

### *2.11 Dynamic Link Library*

A Dynamic Link Library (DLL) is the prototype for component objects. It is a file of code containing functions that can be called from other executable (either an application or another DLL). Programmers use DLLs to provide code that they can reuse and to parcel out. A DLL cannot be run directly. DLL must be called from other executing code [18].

An example would be that Microsoft has a DLL comctl32.dll, which does all the user interface jobs (toolbars, text boxes, scroll bars, etc). Other programs use that DLL so that they will not have to create their own edit box, etc.

Typically, a DLL provides one or more particular functions and a program accesses the functions by creating either a static or dynamic link to the DLL. A static link remains constant during program execution while a dynamic link is created by the program as needed.

A DLL can be used by several applications at the same time. Some DLLs are provided with Windows operating system and available for any Windows application. Other DLLs are written for a particular application and are loaded with the application.

There are a few of advantages of DLL files:

1. Applications would link to this code library, thus saving greatly on duplication of effort and storage space.
2. Applications that used the DLL system would behave exactly the same as all other applications that used it.

# Chapter III

## Development and Implementation

### 3.1 Overview of the Features of the System

This system is characterized by the following aspects:

- ◆ A user-friendly interface that accepts user input data and displays the computed results.
- ◆ A DBMS that stores information, such as crop names, crop growth stages, fertilizers, fertilizer form and fertilizer placement methods.
- ◆ An Excel spreadsheet as a server to hold input data from table queries in the database and to perform pre-calculation for its clients, VB engine and Solver.
- ◆ A Solver application as a server to provide optimization calculation service for Excel on the client side.
- ◆ An OLE Container control as an entrance to various objects available within the operating system that Visual Basic can't handle on its own, including Word documents, and Excel spreadsheets.
- ◆ Application of COM technology to implement IPC, the communication between VB and database adopting OLE DB technology – ADO, the communication between VB and Excel using OLE DB technology – Automation, and the corporation of Excel and Solver add-in relying on VBA. The three main software applications, Excel, Fertilizer database, and Solver used in this project, provide service to each other, commonly accomplishing the functionality of this system.

Figure 4 shows the general picture of this system.



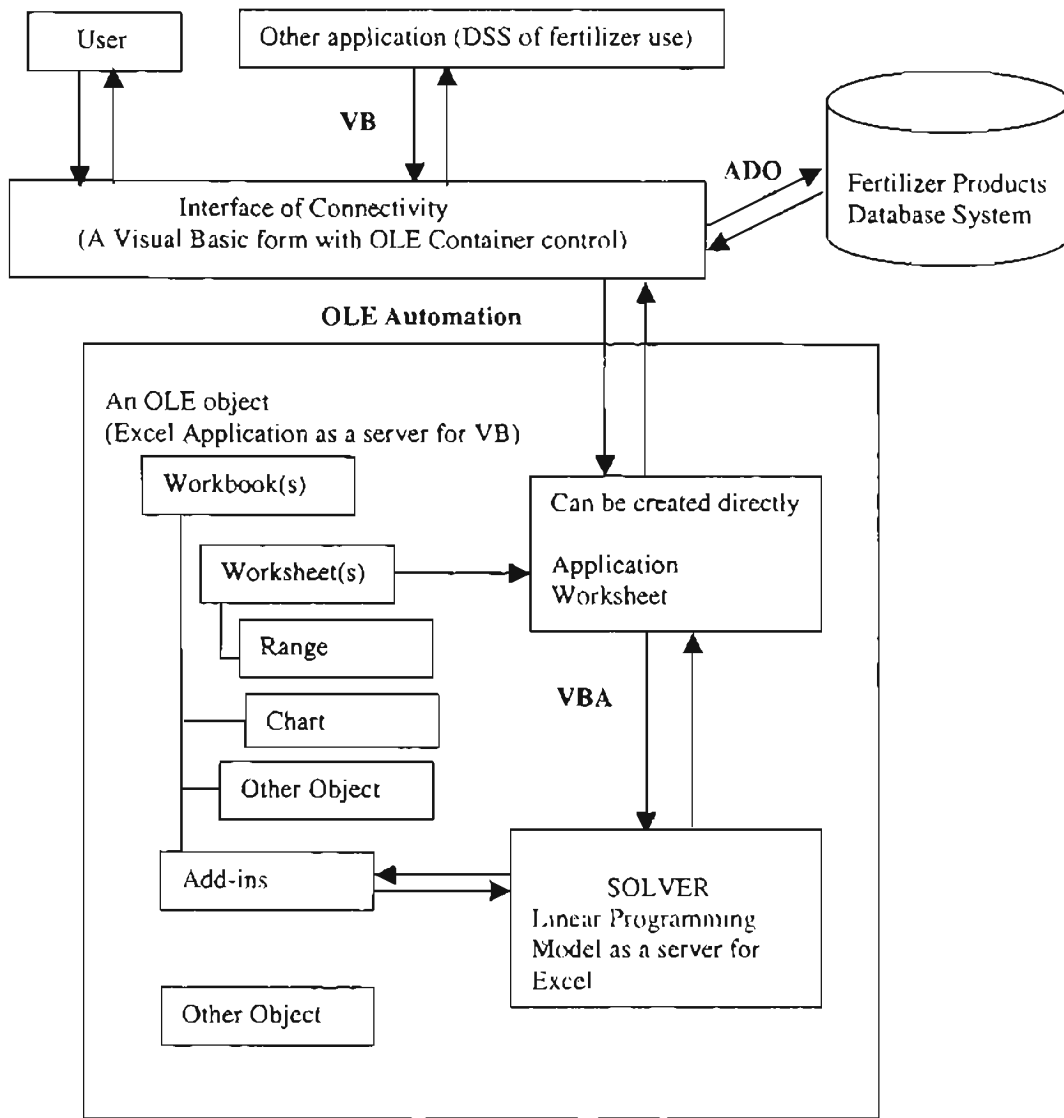


Figure 4. The structure of Optimal Fertilizer Investment System

### 3.2 Design and Implementation

#### 3.2.1 Construct user graphic interface.

The User interface is what appears in this application's window when it runs. It consists of various elements with which a user can interact and control the application. The main part of the user interface is the Form. This is the window displayed at runtime, and it acts as a container for all the elements contained in the interface. There are four

forms created for the interface, each with different components.

**Start Form** This is the first view of this interface, and it functions as a door to this system.

**PictureBox** Shows a picture that symbolizes the application of the software in agriculture.

**Next Command Button** Is used as a switch to the next form.

**Exit Command Button** Allows a user to leave this system.

**PreInformation Form** This form prompts users to input information, including crop name, fertilizer placement phase, and fertilizer placement method, and then it displays corresponding fertilizer form applicable to the fertilizer placement method.

**Crop ListBox** Lists all the names of the crops for user's selection.

**Label 1** Shows the text either selected crop name or unapplied fertilizer code, depending on whether crop name is highlighted or Ok button is clicked.

**Label 2** Shows the text either fertilizer placement method or fertilizer form, depending on whether method name is highlighted or OK button is clicked.

**TextBox1** Displays either crop name when a crop name is highlighted in the Listbox or a unapplied fertilizer code and name on a crop when OK button is clicked.

**TextBox2** Exhibits either method name when method is highlighted in the Method TextBox or fertilizer form (solid or liquid) when OK button is clicked.

**Option Box Array** Lists crop phases, such as Preplant, Panting and Post\_Emergence, to allow user's selection.

**Application Method TextBox** Illustrates corresponding fertilizer placement method when a crop phase is determined.

*ADO Data Control* Is used to access fertilizer database through ADO objects, hidden from the user.

*Back to Start Command Button* Is used to switch back to Start Form.

*Next Command Button* Triggers a switch to the next Form.

*Combination Form* This form allows users to check the fertilizers that are either unavailable or having side-effect on certain crop or both. It shows the groups of fertilizer codes that are unused because they are proper subset of another group, then displays the possible combination of fertilizer codes from the MixTable, that are mutually compatible and safe to mix with other fertilizers.

*Unavailable fertilizer CheckBox* Lists all the fertilizers in the same form. This is a multiple CheckBox that allows a user to check more than one fertilizer names that are currently unavailable on the fertilizer market.

*OK Command Button* Confirms the input in the Unavailable fertilizer CheckBox.

When this button is clicked, the unavailable fertilizers are fixed.

*Unused Combination CheckBox* Shows all the fertilizer combinations, compatible and incompatible, based on the fertilizers stored in the MixTable.

*Possible Combination ListBox* Demonstrates the fertilizer group with all compatible fertilizers.

*ADO Data Control* Is used to access fertilizer database through ADO objects, hidden from the user.

*Back to Start Command Button* Is used to switch back to Start Form.

*Next Command Button* Triggers a switch to the next Form.

*Optimization Form* This form prompts users to input the requirements of N, P, and K, displays the fertilizer names after removing all undesirable fertilizers, and then shows the optimal result.

*N, P, and K Requirement Label* Displays the text of N, P, and K on the Form that a user cannot edit.

*N, P, and K TextBox* Displays the contents of required N, P, and K input by the users, that can be edited.

*OK Command Button* Represents the confirmation action that is carried out when a user clicks the button.

*OLE Control* Is used to insert objects from Microsoft Excel into the VB application, hidden from the user.

*ADO Date Control* Is used to access Fertilizer database through ADO objects, hidden from the user.

*DataGrid control* Displays one or more rows in a table or query from the Fertilizer database. It grants a user the permission to add, delete, or update data in the database.

*User Select Command Button* Allows a user to select the fertilizers.

*Selected Command button* Confirms the selected fertilizers by the user.

*Do Solver Command Button* Drives Solver to compute optimization value.

*Report Picture Box* Displays the computed optimized results in the form of recommendation report, including the optimal mixture of fertilizers and the minimal cost to purchase the mixture.

*Print Command Button* Triggers a print action.

*Back to Start Command Button* Brings a user back to the Starting Form, optional.

*Exit Button* Allows a user to leave this system.

### 3.2.2 Design and implementation of database

This database, named Fertilizer, has relational database structure. This structure is developed emulating the criteria, that are removing data redundancy, keeping data consistent, and improving data accessibility and responsiveness. Furthermore, the selection of fertilizer blending can be the optimum one only by considering all the biological (crop), biochemical (fertilizer/crop), chemical (fertilizer/fertilizer), and physical (fertilizer form/fertilizer placement method). Therefore, the relationship among crop, crop growth period, fertilizer, fertilizer form, and fertilizer placement method must be all concerned when designing this database.

#### *Design of database*

The design of this database goes through following stages:

#### Conceptual database design

This is the process that constructs a model of the information used in an enterprise, independent of all physical consideration [9].

The conceptual design includes:

(1) identifying entities and relationships.

The entities consists of Crop, Method, Fertilizer, and MixTable.

- Crop entity contains the information about crop and unapplied fertilizers on the crop.

- Method entity stores the data related to crop growth phase, fertilizer placement method, and fertilizer form.
- Fertilizer entity includes the data pertaining to fertilizer, such as the contents of N, P, and K in one fertilizer, fertilizer form and fertilizer price.
- MixTable entity stores the information of unmixed fertilizers.

There exist three relationships among these entities.

- a. The relationship between Fertilizer and Crop is one-to-many. One crop cannot be applied by more than one type of fertilizers and one type of fertilizer may not apply on more than one crop.
- b. The relationship between Method and Fertilizer is one-to-many. One form of fertilizer could be applied to different placement methods.
- c. The relationship between Fertilizer and MixTable is one-to-many. One fertilizer may be incompatible with more than one other fertilizers.

(2) Determine attribute domains

#### **Crop table**

- Crop\_Code Crop code, primary key, not null. Text in the range of 1-9999, in ascending order.
- Crop\_Name Crop name, not null. Character in the range of a-z.
- Unapplied\_Fertilizer\_Code Unapplied fertilizer code, foreign key, not null. Text in the range of 1-9999.

#### **Method table**

- Growgh\_Phase\_Code Code for crop growth phase, primary key, not null. Text in the range of 1-9999, in ascending order.

- **Placement\_Method\_Code** Fertilizer placement method code, primary key, not null.  
Text in the range of 1-9999.
- **ApplyMethod** Fertilizer placement method, primary key, not null. Text with 25 characters in the range of a-z.
- **Fertilizer\_Form** Fertilizer form, foreign key, not null. Text with 10 characters in the range of a-z.

### **Fertilizer table**

- **Fertilizer\_Code** Fertilizer code, primary key, not null. Text in the range of 1-9999.
- **Fertilizer\_Name** Fertilizer name, foreign key, not null. Text with 25 characters in the range of a-z.
- **Common\_Name** The alternate names for fertilizers. Text in the range from 1-99 with dash in between.
- **N** The content of Nitrogen in the fertilizer per pound. Decimal number in the range of 0.00-99.00.
- **P** The content of Phosphorous in the fertilizer per pound. Decimal number in the range of 0.00-99.00.
- **K** The content of Potassium in the fertilizer pound. Decimal number in the range of 0.00-99.00.
- **Form** Fertilizer form, foreign key, not null. Text with 10 characters in the range of a-z.
- **Price** Fertilizer price per pound, currency in U.S dollar.

### **MixTable**

- **Fertilizer ID**, primary key, not null. Text in the range of 1-9999.

- Name The fertilizer name, primary key, not null. Text with 25 character in the range of a-z.
- F1 Unmixed fertilizer name. Text with 25 characters in the range of a-z.
- F2 Unmixed fertilizer name. Text with 25 characters in the range of a-z.
- F3 Unmixed fertilizer name. Text with 25 characters in the range of a-z.
- F4 Unmixed fertilizer name. Text with 25 characters in the range of a-z.
- F5 Unmixed fertilizer name. Text with 25 characters in the range of a-z.
- F6 Unmixed fertilizer name. Text with 25 characters in the range of a-z.

### Logical database design

This is the process that constructs a model of the information used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations [9].

The logical design includes mapping the conceptual data model to logical data model, validating the logical model with the technique of normalization, and drawing the Entity-Relation Diagram (E-R Diagram). After logical design, all undesirable features are eliminated from this data model.

#### (1) Normalization.

Normalization is an effective means of ensuring that the models structurally are consistent and logical and have minimal redundancy. Normalization often is executed as a series of steps. Each step corresponds to a specific normal form that has known properties[9]. As normalization proceeds, the relations become progressively more restricted in format, and also less vulnerable to update anomalies [9]. In this project all



tables shown in Figure 5 are in the Third Normal Form after normalization.

Crop Table

<u>Crop Code</u>	Crop_Name	Unapplied_Fertilizer_Code
------------------	-----------	---------------------------

Method Table

<u>Growth Phase</u>	<u>Place Method Code</u>	ApplyMethod	Fertilizer_Form
---------------------	--------------------------	-------------	-----------------

Fertilizer Table

<u>Fertilizer Code</u>	Fertilizer_Name	Common_Name	N_Content	P_Content	K_Content
------------------------	-----------------	-------------	-----------	-----------	-----------

Fertilizer_form	Price
-----------------	-------

MixTable

<u>ID</u>	<u>Fertilizer Name</u>	F1	F2	F3	F4	F5	F6
-----------	------------------------	----	----	----	----	----	----

Figure5. The relations in the Third Normal Form

(2) Draw Entity-Relation Diagram (shown in Figure 6.).

*Establishing and Loading the database*

1) Create Table

- Open Microsoft Access 2000.
- Open blank database.
- Create database name "Fertilizer".
- Create tables using Design View.
- Specify the primary keys and foreign keys in each table.
- Save created tables.

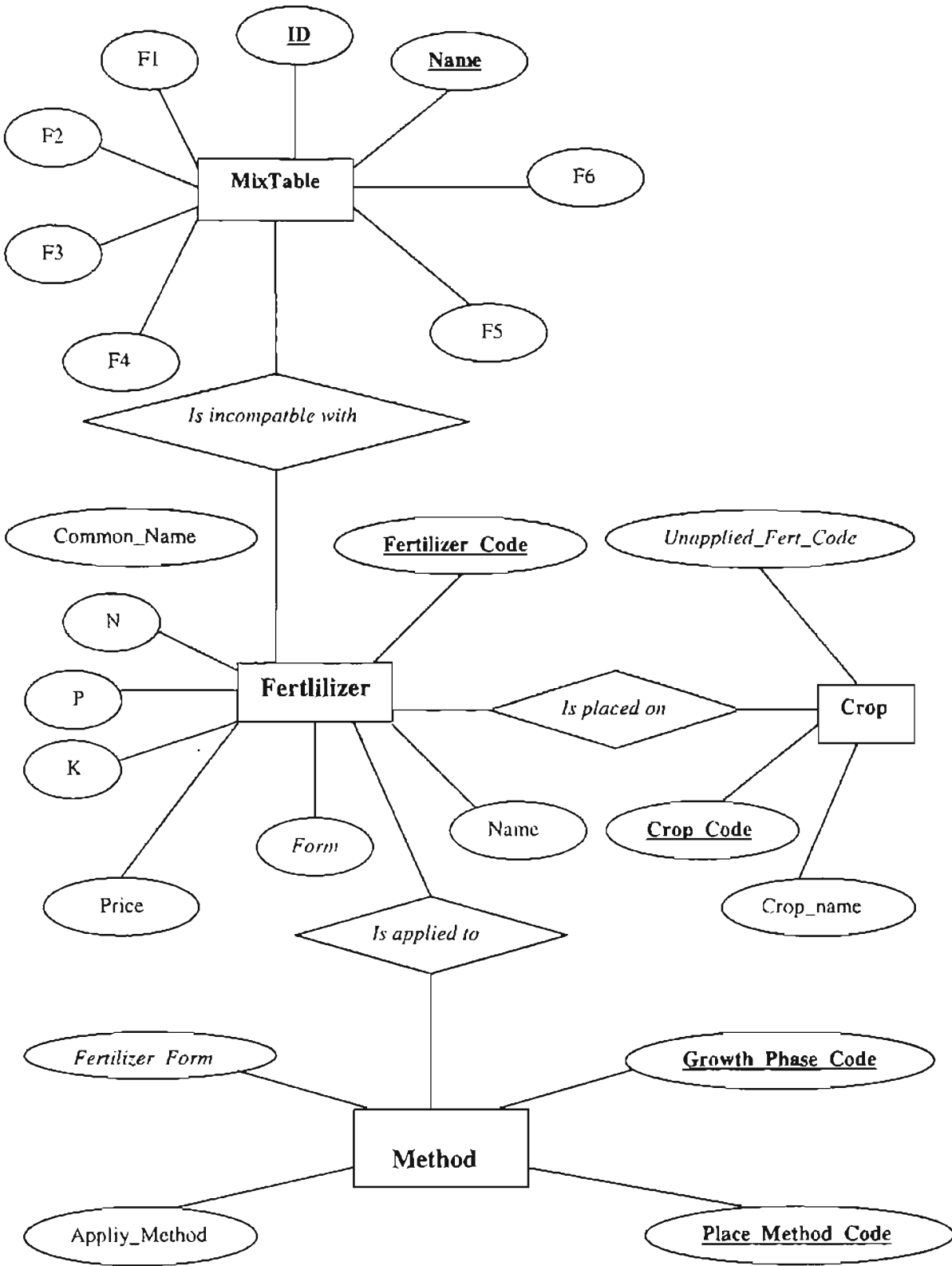


Figure 6. An E-R diagram for the Fertilizer database.

2) Loading date into each table.

Figure 7, 8, 9, 10, and 11 show the tables, Crop, Method, Fertilizer, and Mix Table, respectively.

CropCode	CropName	Unapplied Fertilizer_code
1	Wheat	
10	Potato	06
11	Ryegrass	
12	Tobaco	06
13	Bermudagrass	
14	Peanuts	
15	Soybeans	
16	Mungbeans	
17	Cowpeas	
18	Guar	
19	Small Grains for Grazi	
2	Barley	
20	Legumes in Pasture	
21	Alfalfa	01
22	Sorghum-Sudan Hay	02
23	Garden	
24	Lawn	
25	Native Hay	
26	Hairy Vetch	
27	Other Clover	
28	Millet	

Record: 1 of 36

Figure 7. Crop table.

Growth Phase_Code	Placement Method Code	ApplyMethod	Fertilizer_Form
1	2	Injection	Liquid
1	3	Water-run	Liquid
2	4	Surface banding	Solid
2	5	Subface banding	Solid
3	3	Spay	Liquid
3	6	Side dressing	Solid
3	7	Top dressing	Solid
3	8	Water-run	Liquid

Record: 7 of 9

Figure 8. Method table.

Fertilizer_Code	Name	Common_Nam	N	P	K	Form	Price
01	Ammonium sulphate	21-0-0	0.21	0.00	0.00	Solid	\$0.63
02	Ammonium sulphate nitrate	32-0-0	0.32	0.00	0.00	Solid	\$0.69
03	Urea	46-0-0	0.46	0.00	0.00	Solid	\$0.86
04	Superphosph-Triplephosph	0-46-0	0.00	0.46	0.00	Solid	\$0.61
05	Diammonium phosphate	18-46-0	0.18	0.46	0.00	Solid	\$0.97
06	K-chlonde	0-0-62	0.00	0.00	0.62	Solid	\$0.79
07	Anhydrous Ammonia	82-0-0	0.82	0.00	0.00	Liquid	\$0.84
08	Urea Ammonium Nitrate Solution	32-0-0	0.32	0.00	0.00	Liquid	\$0.49
09	Methylene Ureas	41-0-0	0.41	0.00	0.00	Solid	\$0.78
10	Monoammonium Phosphate	12-51-0	0.12	0.51	0.00	Solid	\$0.95
11	Ammonium Phosohate	11-52-0	0.11	0.52	0.00	Solid	\$0.96
12	Ammonium Phosohate Liquid	10-34-0	0.10	0.34	0.00	Liquid	\$0.59
13	UAN Solution	28-0-0	0.28	0.00	0.00	Liquid	\$0.41
14	Ammonium Polyohosphate Liquid	12-40-0	0.12	0.40	0.00	Liquid	\$0.65
15	Ammonium Phosphate Sulfate	16-20-0	0.16	0.20	0.00	Solid	\$0.94
16	Ammonium Phosphate Nitrate	30-23-0	0.30	0.23	0.00	Solid	\$0.96
17	Granular Monoammonium Phospl	10-52-0	0.10	0.52	0.00	Solid	\$1.04
18	Ammonium Nitrate Liquor	80-0-0	0.80	0.00	0.00	Liquid	\$0.44
19	Ammonium Polyphosphate Solid	11-55-0	0.11	0.55	0.00	Solid	\$1.22
20	Orthophatic Fertilizer Solution	0-25-0	0.00	0.25	0.00	Liquid	\$0.00

Figure 9. Fertilizer Table.

ID	Name	F1	F2	F3	F4	F5	F6
1	Ammonium sulphate			Urea			
2	Ammonium sulphate nitrate			Urea			K-chlonde
3	Urea				Superphosph-Triplephosph		K-chlonde
4	Superphosph-Triplephosph					Diammonium phosphate	
5	Diammonium phosphate						
6	K-chlonde						

Figure 10. MixTable.

### 1.2.3 Algorithm for Selection of Fertilizer Blending

Selecting optimal fertilizer mixture is the goal pursued by this project. To reach this goal, following algorithm is implemented.

1. Determine fertilizer form.

This system prompts a user to input the information about crop growth phase. Based on this input, several corresponding fertilizer placement methods show up. The user selects one among these methods. Both crop growth phase and fertilizer placement method determine the fertilizer form in the method table. By querying this table with the information about crop growth phase and fertilizer application method, the corresponding fertilizer form can be fixed.

2. Retrieve all the fertilizers in the specified form.

The Fertilizer table contains all the fertilizers in either dry or liquid form. Based on the fertilizer form known from the previous steps, a query in the Fertilizer table can provide the user with a group of the fertilizers in the same form.

3. Record inapplicable fertilizers.

By identifying the crop name input by the user, a search is conducted in the crop table for the fertilizer code that has side-effect on the specified crop. If the search is successful, the found fertilizer code is compared with each in the group of fertilizers. If a match occurs, the matched fertilizer code in the group is recorded.

2. Record unavailable and incompatible fertilizers.

Some fertilizers may be unavailable at times, and some may harm the crop when applied on because the chemical reactions occur among incompatible fertilizers. To remove these undesirable fertilizers from the fertilizer group, following steps are taken:

- (1) Prompt the user to check the unavailable fertilizers listed in Unavailable Fertilizer

CheckBox in Combination Form. After the user confirm this action by clicking OK button, these unavailable fertilizers are recorded.

- (2) Compare the code, inapplicable or unavailable, with those in the previous group. Any match leads to the code in the combination being excluded from the previous group. The remaining fertilizers form new group.
- (3) Make all the possible combinations with the fertilizers in the MixTable. These combinations are either compatible or incompatible. Each incompatible combination is indicated by a check mark.
- (4) Remove all the checked combinations and those belonging to the proper subset of some other combination. The left combinations maintain the fertilizers all compatible with one other.
- (5) Concatenate these compatible combinations with the new group, respectively. The set of new combinations are provided to Solver.

#### 3.2.4 Access database from Visual Basic using ADO

Visual Basic 6.0 goes with ADO Data Control that accesses databases through the ADO objects. The ADO Data Control is the door of this system to Fertilizer database.

To access the database through ADO, following properties are set up:

- (1) **ConnectionString** It contains all the information required to connect to Fertilizer database, including:
  - **Provider:** the name of the OLEDB driver that ADO uses to access the Database. For Access database it is Microsoft.Jet.OLE DB. 4.0.
  - **Data Source:** the name of an Access database – the full path to the Access database file (.mdb) where the database is stored. For example, c:\my document\OptimalFert3\Fertilizer.mdb.

- Security Information: Set to True for security, otherwise set to False.

For querying data from the database for Pre Information form, the Connecting String is set as:

```
frmPreInfor.Adodc1.connectionString = "Provider = microsoft.Jet.OLEDB.4.0;Data Source=" & VB.App.Path & "\fertilizer.mdb; Persist security Info = False".
```

(2) RecordSource This represents a database object, such as a table or query, that specifies the rows. In this project SQL statements embedded in VB code are employed to set RecordSource. RecordSource makes up the RecordSet.

For example, to match the crop name input by a user with that stored in Crop table in the database, following SQL statement is used:

```
Adodc1.RecordSource = "SELECT * FROM Crop WHERE CropName = " & "" & Text1.Text & ""
```

(3) RecordSet It is a table or query on which the retrieval, searching, or updating of data can be conducted. For example, to add to the ListBox the name and code of a fertilizer specified by the user, the RecordSet would be set as:

```
FrmCombin.List2.AddItem Adodc1.Recordset.Fields ("Name"). Value + "," + Adodc1.1 Recordset.Fields ("Fertilizer_Code").value
```

### 3.2.5 Displaying of Recordset by DataGrid control

DataGrid control maps an entire table (or Recordset) on a grid that is placed on a VB Form. This control not only displays the Recordset but also allows direct editing of these Recordset, as well as the addition of new rows and the deletion of existing ones. Any change to the field is committed to the database automatically.

For setting DataGrid control several work must be done:

- (1) Placed DataGrid control on the Form
- (2) Set data binding properties
  - Set the DataSource (the name of the ADO Data Control on the Form)
  - Set each column in DataField of the control to their corresponding field names in the database
  - Set allowance for adding, deleting, and updating data in the database
- (3) Build Bookmark for manual selection of fertilizers

### 3.2.6 Realization of OLE Automation

OLE is a way for applications to exchange both data and functions. End users can put together the pieces they need to create their own all-in one software. To incorporate OLE function into the VB application for this project, an OLE container is used. This container hides all the complexity of OLE, and it allows its VB code to directly manipulate the objects in Excel, such as cells, range, worksheet, functions, and formula.

(1) To link objects in an OLE Container control, following steps are taken:

- a. Place a new OLE Container control on the frmSolver
- b. Select *Create New Object* option in the property window of OLE Container Control.
- c. Create OLE object with VB code.

For example, to create an Excel application as a object of OLE control, following statement is used:

```
Set obj1 = CreateObject ("Excel.application")
```



d. Open an existing Excel file with VB code.

For instance, the VB statement:

```
Obj1.Workbooks.Open Filename: = VB.App.Path & "\Fertilizer1.xls"
```

is used to open the existing Excel file that contains Solver and Macro.

(2) Using VB code manipulate range object, cell object, SUMPRODUCT, constraints, objective function, and variable cells that used to be manual operations. The OLE object obj1 can only host the objects listed in Excel.

Because Solver is an Excel add-in rather than an Excel object, OLE cannot control Solver. Therefore, this application has to resort to an Excel spreadsheet as interface on which Solver can work. Furthermore, the data retrieved from database must be written into Excel spreadsheet. This is the job of VB code.

For instance, to write to Excel spreadsheet the requirements of N, P, and K from the user, the VB code can be:

```
.Range ( "B3").Value = Text2(0).Text
```

```
.Range ( "B4").Value = Text2(1).Text
```

```
.Range ( "B5").Value = Text2(2).Text
```

```
.Range ( "B7").Value = "Amount of fertilizer"
```

To write to the Excel spreadsheet the contents of N, P, and K in a fertilizer and their prices stored in the database,

the VB statements are:

```
.Range ( "E3" ).Value = frmPreInfor.Adodc1.Adodc1.Recordset.fields("N").Value
```

```
.Range ( "E4" ).Value = frmPreInfor.Adodc1.Adodc1.Recordset.fields("P").Value
```

```
.Range ( "E5" ).Value = frmPreInfor.Adodc1.Adodc1.Recordset.fields("K").Value
```

*.Range ( "E8" ).Value =fmPreInfor.Adodc1.Adodc1.Recordset.fields("Price").Value*

To calculate the optimal result based on the contents of N, P, and K and the prices of the fertilizers, using SUMPRODUCT function in Excel, following VB statements are used:

*.Range("C3").Value = SUMPRODUCT("E3:H3", "E7:H7")*

*.Range("C4").Value = SUMPRODUCT("E4:H4", "E7:H7")*

*.Range("C5").Value = SUMPRODUCT("E5:H5", "E7:H7")*

However VB code is unable to drive solver directly, therefore another programming language, VBA embedded in Excel, is employed.

### 3.2.7 Implementation of Solver with VBA

OLE Automaton allows manipulation of objects exposed by another application from what within user's Visual Basic for applications (VBA).

In this project, the interface accepts from a user the inputs, such as the requirements of N, P, and K, and then places into Excel spreadsheet the table query results, such as the contents of N, P and K as well as the prices of the fertilizers. After setting the positions of the inputs as well as the formula for objective function and constraint equations,

Visual Basic calls Excel VBA functions to drive Solver for computation.

Solver add-ins is a member of Dynamic Link Libraries and is used to extend the functionality in Excel. To use Solver add-in functions in a VBA (VBA macro), a reference is in advance set up to the Solver add-in, located in /office/Solver/folder. The path for the reference is: *manual bar\micro\VB project\reference\browser.*

From VBA program, five Solver DLLs, SolverReset, SolverAdd, SolverOK, SolverSolve, and SolverFinish, are called to solve the optimal model.

**SolverReset** function erases all cell selections and constraints from Solver Parameters dialog box and restores all settings in the Solver dialog box to their defaults. This function is equivalent to choosing Solver from the Tools menu and choosing the Reset All button in the Solver Parameters dialog box. The syntax for the SolverReset function is:

SolverReset()

**SolverAdd** function adds a constraint to the current problem. This function is equivalent to choosing Solver from the Tools menu and choosing the Add button in the Solver Parameters dialog box. The syntax for the SolverAdd function is:

SolverAdd(*CellRef*:=, *Relation*:=, *FormulaText*:=)

*CellRef*. Is a reference to a cell or a range of cells on the active worksheet and forms the left hand side of the constraint.

*Relation*. Specifies the arithmetic relationship between the left and right sides, or whether *CellRef*. must have an integer value at the solution.

Relation	Relationship
1	<=
2	=
3	>=
4	Int (CellRef is an integer variable)

*FormulaText*. Is the right hand side of the constraint and will often be a single number,

but it may be a formula or a reference to a range of cells.

**SolverOK** function defines a basic Solver model. This function is equivalent to clicking Solver on the Tools menu and then specifying options in the Solver Parameters dialog box. The syntax for the SolverOK function is:

**SolverOK** ( *SetCell*, *MaxMinVal*, *ValueOf*, *ByChange* )

**SetCell**. Specifies the target cell.

**MaxMinVal**. Corresponds to whether or not a user wants to solve the target cell for a maximum value (1), a minimum value (2), or a specific value (3).

**ValueOf**. Specifies the value to which the target cell is matched.

**Bychange**. Specifies the cell or range of cells that will be changed.

**SolverSolve** function solves the model using the parameters specified with SolverOK.

Executing the SolverSolve function is equivalent to clicking Solve in the Solver Parameters Dialog Box. The syntax for the SolverSolve function is:

**SolverSolve**(*UserFinish*, *ShowRef*)

**UserFinish**. Indicates whether or not the user want to finish solving the model. To return the result without displaying the Solver Results dialog box, set this argument to TRUE, otherwise set this argument to FALSE.

**ShowRef**. Identifies the macro that is called when Solver returns an intermediate solution. The ShowRef argument should be used only when TRUE is passed to the StepThru argument of the SolverOptions function.

**SolverFinish** function indicates what to do with the final results and what kind of report to create once the solution process is completed. The syntax for the SolverFinish function is:

SolverFinish ( *KeepFinal*, *ReportArray*)

**KeepFinal.** Indicates what to do with the final results. If keepFinal is 1, the final solution values are kept in the changing cells, replacing the values. If KeepFinal is 2, the final solution values are discarded, and the former values are restored.

**ReportArray.** Specifies an array which indicates the type of report Microsoft Excel will create when the solution is reached. If ReportArray is set to 1, Microsoft Excel creates an Answer Report. If set to 2, Microsoft Excel creates a Sensitivity Report, and if set to 3, Microsoft Excel creates a Limits Report.

This process from setting spreadsheet to computing optimal value through Solver is quite difficult and complicate. However, with the help of OLE Automation this process goes automatic and unperceived.

# Chapter IV

## Results

Following steps will lead to an optimal solution:

1. Enter Starting Form.

- a. Click on TrySolver1.exe to enter the Optimal Fertilizer Investment System.
- b. A picture shows up. This picture symbolizes the application of computer software in agriculture.
- c. Click on the Next button to switch to PreInformation Form.

( Starting Form is shown in Figure 11.).

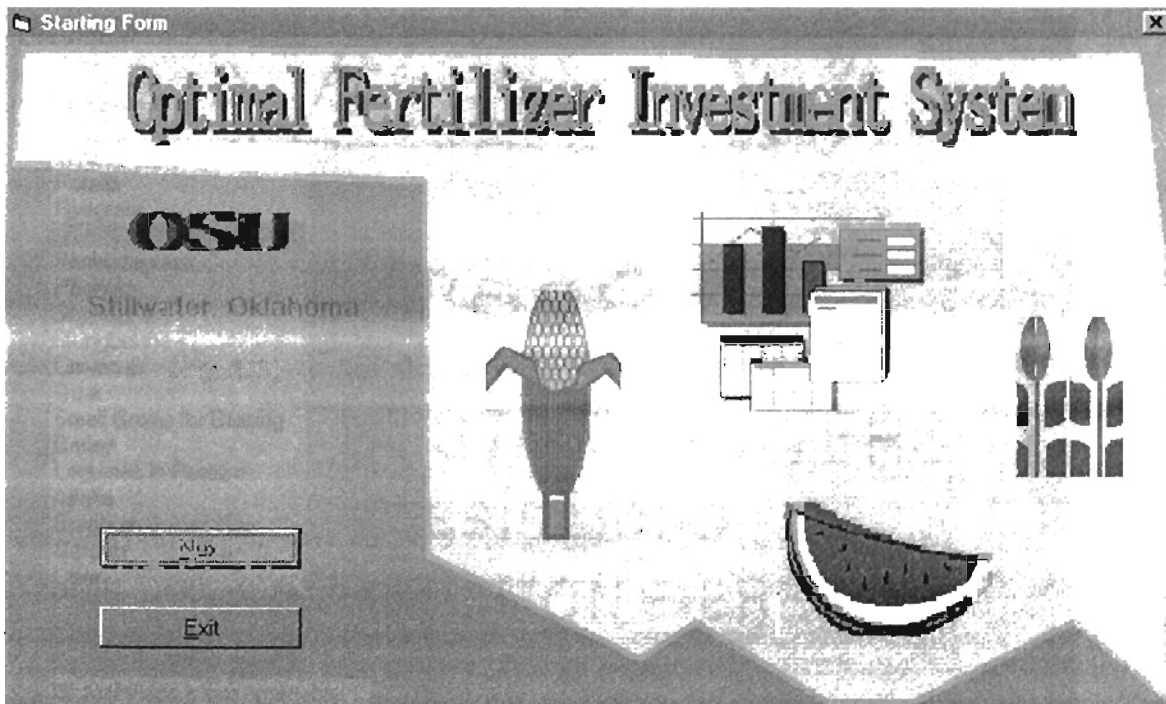


Figure 11. Starting Form.

2. Work in PreInformation Form.

- a. Select one crop by highlighting the crop name in the Crop Selecting ListBox, using the scroll bar when needed.

- b. The selected crop name appears in the Unapplied Fertilizer Code TextBox.
- c. Click on OK button, and then the unapplied fertilizer code on the specified crop are shown in the Unapplied Fertilizer Code TextBox.
- d. Choose one from fertilizer application phases, Pre-Plant, At Planting, and Post-Emergence, in the Operation Box Array.
- e. The corresponding fertilizer placement method is displayed in the Application Methods TextBox.
- f. Select one method from fertilizer placement methods by highlighting the method title.
- g. The selected method title can be seen in the Fertilizer Form TextBox.

The screenshot shows a software window titled "PreInformation" with the following components:

- Crop Selecting List:** A scrollable list of crops including Wheat, Potato, Ryegrass, Tobacco, Bermudagrass, Peanuts, Soybeans, Mungbeans, Cowpeas, Guar, Small Grains for Grazing, Barley, Legumes in Pasture, Alfalfa, Sorghum-Sudan Hay, Garden, and Lawn. "Tobacco" is currently selected.
- Application Phase:** Three radio button options: "Pre-Plant" (selected), "At Planting", and "Post Emergence".
- Application Methods:** A list box containing "Broadcast", "Injection", and "Water-run". "Broadcast" is highlighted.
- Fertilizer Form:** A text box containing the word "Solid".
- Unapplied Fertilizer Code:** A text box at the bottom left containing the text "06 K-chloride is not applicable".
- Buttons:** "Done" (bottom left), "Back to Start" (bottom center), and "Next" (bottom right, highlighted with a dashed border).

Figure 12. PreInformation Form.

h. Click on OK button, and then the corresponding fertilizer form are shown in the Fertilizer Form TextBox.

i. Click on Next button to switch to Combination Form.

(PreInformation Form is shown in Figure 12).

### 3. Manipulate in Combination Form.

a. Indicate one or more fertilizers, either unavailable or undesirable, by checking in the box on the left of the fertilizer name in the Select Unavailable Fertilizer CheckBox, using scroll bar when needed.

b. Click on OK button to confirm the “indication” operation.

c. A group of combinations of fertilizer codes is shown in the Select Unavailable Fertilizer CheckBox. They are either with the check mark in the left check box or not. The checked combinations having incompatible fertilizers within them are removed from the set of combinations. The unchecked combinations contain all the fertilizer compatible with each other.

d. The unchecked combinations are displayed in the Possible Combination TextBox. Each of which is respectively concatenated with the other safe fertilizers to form new combinations that are provided to Solver, respectively.

e. Click on Next button to switch to Optimization of Fertilizer Investment Form.

(Combination Table is shown in Figure 13).

### 4. Operate in Optimization of Fertilizer Investment Form.

a. Enter the requirements of N, P and K in the corresponding TextBox.

b. Click on OK button to confirm the requirement inputs.



**Combination**

**Select Unavailable Fertilizers**

- Ammonium Nitrate,45
- Ammonium Phosphate,11
- Ammonium Phosphate Nitrate,16
- Ammonium Phosphate Sulfate,15
- Ammonium Polyphosphate Solid,19
- Ammonium sulphate nitrate,02
- Ammonium sulphate,01
- Diammonium phosphate,05
- Eagle, 43
- Fertilier with High Potassium,37

Find Compatible Combinations

Check Unused Combinations	Possible Combinations
<input checked="" type="checkbox"/> 05	03,05
<input checked="" type="checkbox"/> 06	04,05
<input checked="" type="checkbox"/> 01,02	01,02,04
<input checked="" type="checkbox"/> 01,04	01,02,05
<input checked="" type="checkbox"/> 01,05	
<input checked="" type="checkbox"/> 01,06	
<input checked="" type="checkbox"/> 02,04	
<input checked="" type="checkbox"/> 02,05	

Back to Start      Next Form

Figure 13. Combination Form.

- c. Click the User Select Mode button to change from automatic selection mode to user selection mode. This operation is optional.
- d. Choose one or more preferable fertilizers from the TextBox, Fertilizer Information Selection and Add Table, using the scroll bar when needed. This operation is optional.
- e. Click on Selected button to confirm the chosen fertilizers. This is an optional operation.

	A	B	C	D	E	F	G	H
2		NPK need	NPK got		Ammonium Nitrate	Urea	Methylene Ureas	Monoammonium P
3	N	24	24		0.34	0.46	0.41	0.
4	P	32	32		0	0	0	0.
5	K	42	42		0	0	0	
6								
7			Amount of fertilizer		0	0	0	4.9180291
8			Price of fertilizer		\$0.61	\$0.86	\$0.78	\$0.95
9								
10			Total Cost		145.3831056			
11								
12		NPK need	NPK got		Ammonium Nitrate	Methylene Ureas	Monoammonium Phos	Ammonium Phosol
13	N	24	24		0.34	0.41	0.12	0
14	P	32	32		0	0	0.51	0
15	K	42	42		0	0	0	
16								
17			Amount of fertilizer		35.67775064	0	0	
18			Price of fertilizer		\$0.61	\$0.78	\$0.95	\$0.95
19								
20			Total Cost		142.7199482			
21								
22		NPK need	NPK got		Ammonium Nitrate	Methylene Ureas	Monoammonium Phos	Ammonium Phosol

Figure 14. The optimal results on the Spreadsheet.

f. Click on Do Solver button to trigger Solver for optimal computation. Figure 14 shows the computed results written on Excel spreadsheet after using Solver:

- Individual fertilizer name, contents of N, P, and K, and price in each combination.
- The required contents of N, P and K for reaching a yield goal.
- The contents of N, P, and K obtained from each combination after applying Solver.
- The total cost for satisfying the requirements of N, P, and K in each combination after running Solver.

g. The optimal solution is displayed in the TextBox, Report of Optimization for Fertilizer Purchase. This solution includes the fertilizer names with the needed pounds of the fertilizers in the blending and the total cost for this blending.

- h. Click on the Print Report button to get a written report.
- i. Click on the Restart button to return to the Starting Form. This operation is optional.
- j. Click on the Exit Application button to leave this system.

(Optimization of Fertilizer Investment Form is shown in Figure15.).

**Optimization of Fertilizer Investment**

**Fertilizer Information and Records Adding, Deleting, and Updating Table**

Fertilizer Code	Name	Common Name	N	P	K	Form	Price
01	Ammonium sulphate	21-0-0	21.0%	0.0%	0.0%	Solid	\$0.63
02	Ammonium sulphate nitrate	32-0-0	32.0%	0.0%	0.0%	Solid	\$0.69
03	Urea	46-0-0	46.0%	0.0%	0.0%	Solid	\$0.86
04	Superphosph-Triplephosph	0-46-0	0.0%	46.0%	0.0%	Solid	\$0.61
05	Diammonium phosphate	18-46-0	18.0%	46.0%	0.0%	Solid	\$0.97
06	K chloride	0-0-62	0.0%	0.0%	62.0%	Solid	\$0.79
07	Anhydrous Ammonia	82-0-0	82.0%	0.0%	0.0%	Liquid	\$0.84
08	Urea Ammonium Nitrate Solution	32-0-0	32.0%	0.0%	0.0%	Liquid	\$0.49

**Report of Optimization for Fertilizer Purchase**

N, P, K requirements

N:

P:

K:

The optimal blending of fertilizers is:

Ammonium Nitrate 36 lbs  
 Potassium Nitrate 91 lbs  
 Superphosph-Triplephosph 70 lbs

with minimal total cost \$142.72

Figure15. The Optimization of Fertilizer Investment Form.

# Chapter V

## Conclusions and Future Work

### 5.1 Summary

This study builds an Optimal Fertilizer Investment System to assist farmers in determining fertilizer investment. This system consists of a relational database to store fertilizer information, an Excel spreadsheet to hold the data retrieved from the database, a computing tool, Solver, to calculate optimal solution, and a user graphic interface to allow a user interactive with this system.

The function of this system depends on some technologies, algorithm and programming languages.

- ADO offers a simple, high-level view of the Fertilizer database. Its basic objects help establish a connection to this database, execute commands against the database, and retrieve information from the database. In this project ADO functions as a bridge between Visual Basic and Fertilizer database, making the access to database through VB possible.
- OLE Automation connects Visual Basic to Excel spreadsheet. This connection contributes to the exposure of Excel objects to VB application and makes it feasible while working in VB environment, using Excel functions to directly manipulate spreadsheet.
- The algorithm for finding the compatible fertilizer combinations is crucial to the implementation of this system. The algorithm is based on the biological, biochemical, chemical, and physical considerations of the fertilizers and tries to make all possible

fertilizer combinations with no fertilizer and no compatible situation being ignored. As the algorithm proceeds, the fertilizers in each combination become progressively more restricted in compatibility. After removing all the combinations with the fertilizers, inapplicable, unavailable, undesirable, and incompatible, the remaining combinations contain all the fertilizers compatible one another.

- Visual Basic is critical for this system. It runs through the process, directly or indirectly controlling or participating the execution of each software application in this system.
- Visual Basic for Applications successfully fills the gap between VB and Excel Solver. Substituting for VB, it effectively drives Solver for an optimal solution.

## **5.2 Conclusions**

- This project provides the users with an optimal solution for fertilizer investment and frees users from the difficult tool operation. Users interact with this system only through a user graphic interface, and simple keystrokes and mouse clicks can help them to gain an optimal result.
- This project realizes the interprocess communications with COM technology. Visual Basic, Fertilizer database, and Excel are mutually independent software applications. With COM technology, these software are integrated into a complete and integrate application. Each component of this application provides to and receives from each other the services as COM objects. They communicate and cooperate, commonly satisfying the requirements for this system.

### **5.3 Future Work**

This system was designed as a desktop application to run on a Windows operating system with MS Office installed. In the future, a new feature, using either ActiveX Controls or Java Applets, should be added to this system so that wherever a user is able to use this system through a web browser, such as Netscape or Microsoft IE. The feature will allow more users to benefit from this system.

## BIBLIOGRAPHY

- [1] Aaron, Bud., Thompson, Ben. *ActiveX*. Schoolcraft, MI: Prima Communications, Inc., 1996.
- [2] Appleman, Dan. *Dan Appleman's Developing COM/ActiveX Components with Visual Basic 6*. Indianapolis, IN: Sams.net, 1999.
- [3] Black, Charles A. *Soil Fertility Evaluation and Control*. Boca Raton, Florida: Lewis Publisher, 1993.
- [4] Box, Don. *Essential COM*. Reading, Mass.: Addison Wesley, 1998.
- [5] Chappell, D. *ActiveX OLE A Guide for Developers & Managers*. Washington: Microsoft Press, 1996.
- [6] Chappell, D. *Understanding ActiveX and OLE*. Washington: Microsoft Press, 1996.
- [7] Collings, Gilbert H. *Commercial Fertilizers Their Sources and Use*. New York: McGraw-Hill Book Company, Inc., 1990.
- [8] Connolly, Thomas M. *Database Systems: A Practical Approach to Design, Implementation and Management*. Harlow, England; Reading, Mass: Addison-Wesley, c1999.
- [9] Ernst, W., Kottler John J. *Presenting ActiveX*. Indianapolis, IN: Sams.net, 1996.
- [10] Gass, Saul I. *Linear Programming Methods and Applications*. New York: McGraw – Hill Book Company, 1969.
- [11] Gary T., Leavens, Murali. *Foundations of Component-based Systems*. Cambridge [England]; New York: Cambridge University Press, 2000.
- [12] Getz, Ken. *VBA Developer's Handbook*. San Francisco: Sybex, c1997.
- [13] Holzner, Steven. *ADO Programming in Visual Basic 6*. Upper Saddle River, NJ: Prentice Hall PTR, 2000.
- [14] [http://cs.nyu.edu/faculty/overton/g22\\_lp/encyc/article\\_web.html](http://cs.nyu.edu/faculty/overton/g22_lp/encyc/article_web.html)
- [15] <http://msdn.microsoft.com/library/office97/ORKht/036.htm#ORK036C1>

- [16] <http://solo.abac.com/dllarchive/define.html>
- [17] <http://www.fertilizer.org/PUBLISH/PUBMAN/introd2.html>
- [18] <http://www.jics.cs.vtk.edu/EXCEL/excel.html>
- [19] <http://www.microsoft.com/TechNet/VBA/ProdFact/vbaprim.asp>
- [20] <http://www.vbexplorer.com/wrox/vbcpmsamp.asp>
- [21] Jones, Walken B. *Excel for Windows 95 Power Programming with VBA*. Forster City, CA:IDG Books Worldwide, 1996.
- [22] Jones, Ulysses. *Fertilizers and Soil Fertility*. Reston, Virginia: Prentice-Hall Company, 1982.
- [23] Lamport, Leslie. *On Interprocess Communication*. Palo Alto, California: Digital System Research Center, 1986.
- [24] Laney, Jeff. *ActiveX and COM – Part I*. Developer Zone: National Instruments, 2000.
- [25] Litwin, Paul. *Microsoft ActiveX Data Objects (ADO) Programming*. <http://www.microsoft.com/accessdev/article/movs202.html>, 2000.
- [26] Lomax, Paul., *VB & VBA in a Nutshell: The Language*. Sebastopol, CA: O'Reilly & Associates, Inc., 1998.
- [27] Lu, H. *COMSC 5423 Lecture Notes*. Stillwater, OK: Oklahoma State University, 1998.
- [28] Marshall, Donis. *ActiveX/OLE Programming : Building Stable Components with Microsoft Foundation Class*. Gilroy, CA : CMP Books, 1998.
- [29] Microsoft Corporation. *About Interprocess Communications*. Win32 Software Development Kit: Microsoft Corporation, 1992-1995.
- [30] Microsoft Corporation. *Automation Programmer's Reference: Using ActiveX*. Washington: Microsoft Press, 1997.
- [31] Mojica, Jose. *ActiveX controls with Visual Basic 5.0*. Forster City, CA: IDG Books Worldwide, 1997.
- [32] Murty, Katta G. *Linear Programming*. New York: John Wiley & Sons, Inc, 1983.



- [33] Neou, Vivian. *ActiveX to Go*. New York: Simon & Schuster Trade, 1997.
- [34] Nering, Evar D., Tucker, Albert W. *Computer Science and Scientific Computing Linear Programs and Relate Problem*. San Diego, CA: Academic Press, Inc, 1993.
- [35] Novalis, Susan. *Access 2000 VBA Handbook*. San Francisco: SYBEX, 1999.
- [36] Petroustos, E. *Mastering Visual Basic 6.0*. Alameda, CA: SYBEX Inc., 1999.
- [37] Ravindran, A., Don, Phillips T., James, Solberg J. *Operations Research Principles and Practice*. New York: John Wiley & Sons, 1987.
- [38] Platt, David S. *The Essence of COM with Active X A Programmer's Workbook*. Reston, VA: Prentice - Hall, PTR, 1998.
- [39] Ren, J. *A Decision Support System for Fertilizer Use* (Master's Thesis for the Department of Computer Science). Stillwater, OK: Oklahoma State University, 2000.
- [40] Roff, Jason T. Petrusha, Ron. *ADO: ActiveX Data Objects*. Sebastopol, CA: O'Reilly & Associates, Inc., 2001.
- [41] Rogerson, Dale. *Inside COM: Microsoft's Component Object Model*. Redmond: Microsoft Press, 1997.
- [42] Shelly, Gary B. *Microsoft Office 97: Advanced Concepts and Techniques*. Mass.: Cambridge, Course Technology, 1999.
- [43] Siberschatz, A., Korth, H.F., Sudarshan, S. *Database System Concepts*. New York: McGraw-Hill. Inc., 1997.
- [44] Soil Improvement Committee California Fertilizer Association. *Western Fertilizer Handbook*. Danville, IL: Interstate Publishers, Inc., 1985.
- [45] Stockton, R.S. *Introduction to Linear Programming*. Homewood, IL: Richard D. Irwin, Inc., 1971.
- [46] Swanson, Leonard W. *Linear Programming Basic Theory and Applications*. McGraw-Hill: Book Company, 1980.
- [47] Taylor, Harold H. *Fertilizer Use and Price Statistics, 1960 – 1993*. Herndon, VA: United State, Department of Agriculture, 1994.

- [48] Taylor, Harold H. *Price Prospects for Major Primary Commodities 1990 – 2005. Agricultural Products Fertilizers Tropical Timber, volume II.* Washington, DC: The World Bank, 1990.
- [49] Ullman, J.D. *Principles of Database and Knowledge-base System.* Rockville, MD: Computer Science Press, c1988-c1989.
- [50] Whigham, David. *Quantitative Business Methods Using Excel.* Oxford, New York: Oxford University Press, 1998.
- [51] Esposito, D. *With Further ADO: Coding Active Data Object 2.0 with Visual Studio 6.0.* Microsoft System Journal, 14(2):17-32,1999.
- [52] Esposito, D. *Exposing Your Custom Data in a Standard Way Through ADO and OLE DB.* Microsoft System Journal, 14(6): 35-50,1999.
- [53] Box, D., Brow, K., Ewald, T. J., Sells, C. *Effective COM Programming: Seven Tips for Building Better COM-based Applications.* Microsoft System Journal, 11(5): 63-80, 1996.
- [54] Rauch, S. *Talk to Any Database the COM Way Using the OLE DB Interface.* Microsoft System Journal, Microsoft Inc., 11(7): 19-32,1996.
- [55] Stanley, J.T. *Extend Office 2000 Applications with Custom-Coded Add-Ins.* Inside Microsoft Visual Basic, Microsoft Inc., August, 2000.
- [56] Hopkins, S. *Connect Objects.* Visual Basic Programmer's Journal, Microsoft Inc., 10(9), 2000.
- [57] Schultes, S. *Use SOL with ADO.* Visual Basic Programmer's Journal, Microsoft Inc., 10(7), 2000.
- [58] Vaughn, W.R. *ADO Command Strategies.* Visual Basic Developer, Microsoft Inc., October, 2000.
- [59] Roberts, B. *An ADO Command Factory for Stored Procedures.* Smart Access, Microsoft Inc., pp. 26-38, September, 2000.
- [60] Vogel, P. *COM Add-ins in Detail.* Microsoft Office & Visual Basic For Applications Developer, April, 2000.

# APPENDIXES

## Appendix A. Acronyms and Abbreviations

<b>Acronym</b>	<b>Explanation</b>
ADO	ActiveX Data Object
API	Application Programming Interface
COM	Component Object Model
DBMS	Database Management System
DDE	Dynamic Data Exchange
DDL	Data Definition Language
DLL	Dynamic Link Libraries
DML	Data Manipulation Language
EXEs	Executable
GUI	Graphic User Interface
IDE	Integrated Development Environment
IPC	Interprocess Communications
LP	Linear Programming
MS	Microsoft
OCX	OLE Controls
OLE	Object Link Embedded
OLE DB	Object Link Embedded Database
OSU	Oklahoma State University
RDBMS	Relational Database Management System
RPC	Remote Process Call

SFUDSS	Soil Fertilizer Use Decision Support System
SQL	Structured Query Language
VB	Visual Basic
VBA	Visual Basic for Applications

## **Appendix B. GLOSSARY**

**ActiveX Control** A control using ActiveX technologies that can be downloaded and executed by a Web browser. ActiveX is a set of rules for how applications should share information. Programmers can develop ActiveX controls in a variety of languages, including C, C++, Visual Basic, and Java. ActiveX controls have full access to the Windows operating system and are currently limited to Windows environments.

**ActiveX Data Object (ADO)** A Microsoft's newest high-level interface for data objects. ADO can be used to access all sorts of different types of data, including web pages, spreadsheets, and other types of documents. Together with OLE DB and ODBC, ADO is one of the main components of Microsoft's Universal Data Access (UDA) specification, which is designed to provide a consistent way of accessing data regardless of how the data is structured.

**Add-in** A software program that extends the capabilities of larger programs. For example, there are many Excel add-ins designed to complement the basic functionality offered by Excel. In the Windows environment, add-ins is becoming increasingly common thanks to OLE 2.0.

**ADO Data Control** A form of an ActiveX controls that represents the ActiveX Data Objects (ADO) and accesses database through the ADO objects. ADO Data Control is placed on a Form and functions as a user application's visual gateway to a database. A user can set it up to "see" any table or query in a database with point-and-click operations.

**Application Program Interface (API)** An abbreviation of application program interface, a set of routines, protocols, and tools for building software applications. API

makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together. Most operating environments, such as MS-Windows, provide an API so that programmers can write applications consistent with the operating environment.

**Component Object Model (COM)** A model for binary code developed by Microsoft that enables programmers to develop objects that can be accessed by any COM-compliant application. Both OLE and ActiveX are based on COM.

**Control** (1) An object in a window or dialog box. Examples of controls include push-buttons, scroll bars, radio buttons, and pull-down menus. (2) An OLE or ActiveX object.

**Data Definition Language (DDL)** A descriptive language that allows the Database Administrator or user to describe and name the entities required for the application and the relationships that may exist between the different entities.

**Data Manipulation Language (DML)** A language that provides a set of operations to support the basic data manipulation operations on the data held in the database. Data manipulation operations usually include insertion of new data into the database, the modification of data stored in the database, the retrieval of data contained in the database, and the deletion of data from the database.

**Database** A shared collection of logically related data (and a description of this data), designed to meet the information needs of an organization.

**DataGrid Control** A form of an ActiveX control that comes with Visual Basic and can be used to map an entire table or queries. DataGrid control allows direct editing of its cells, as well as the addition of new rows and the deletion of existing ones.

**Dynamic Link Libraries (DLL)** A library of executable functions or data that can be used by a Windows application. A DLL provides one or more particular functions and a program accesses the functions by creating either a static or dynamic link to the DLL. A static link remains constant during program execution while an dynamic link is created by the program as needed. DLLs can also contain just data. DLL files usually end with the extension .dll,.exe., drv, or .fon. A DLL can be used by several applications at the same time.

**First Normal Form** The first state of tables in the normalization for table design. In the First Normal Form, there is exactly one value in the section of each row and column.

**Interprocess Communication (IPC)** A capability supported by some operating systems that allows one process to communicate with another process. The processes can be running on the same computer or on different computers connected through a network. IPC enables one application to control another application, and for several applications to share the same data without interfering with one another. IPC is required in all multiprocessing systems.

**Linear Programming (LP)** A standard tool that allocates a finite set of resources in an optimal way. Linear program is specific class of mathematical problems, in which a linear function is either maximized or minimized subject to given linear constraints. This problem class is broad enough to encompass many interesting and important applications yet, specific enough to be tractable even if the number of variable is large.

**Microsoft Access** A typical PC-based DBMS that is capable of storing, sorting, and retrieving data for a variety of applications. This DBMS package provides the tools to create tables, queries, forms, and reports, and to develop customized database

applications using the Microsoft Access micro language or the Microsoft Visual Basic for Application language. Microsoft Access can be used as a standalone system on a single PC or as a multi-user system on a PC network.

**Microsoft Excel** A business application for accounting and financial calculations, however, also powerful to perform many of calculations that scientists and engineers use. Excel has sufficient numeric precision to represent most of the values in calculations of interest to those in technical fields. There are also facilities for creating user-defined functions and automating often-repeated task/calculations with macros. Excel gains its popularity due to this feature that makes information easy to present in meaningful way.

**Object Link Embedded (OLE)** An abbreviation of Object Linking and Embedding that is a compound document standard developed by Microsoft. It enables users to create objects with one application and then link or embed them in a second application. Embedded objects retain their original format and links to the application that created them.

**Object Link Embedding for Database (OLE DB)** A set of data objects defined by Microsoft that allows OLE-oriented applications to share and manipulate sets of data as objects. OLE DB provides access to any data source, including relational and non-relational database, e-mail, file systems, text, graphics, custom business objects, and more. OLE DB is an object-oriented specification based on a C++ API. As components can be thought of as secure, reusable object, components can be retrieved as both data consumers and data providers at the same time. Consumers take data from OLE DB interfaces and providers expose OLE DB interfaces.



**Relational Database Management System (RDBMS)** A type of database management system (DBMS) that stores data in the form of related tables. Relational databases are powerful because they require few assumptions about how data is related or how it will be extracted from the database. As a result, the same database can be viewed in many different ways. An important feature of relational systems is that a single database can be spread across several tables. This differs from flat-file databases, in which each database is self-contained in a single table. Almost all full-scale database systems are RDBMS's.

**Second Normal Form** The second state of tables in the normalization in the table design. In the Second Normal Form, every non-primary key is fully and functionally dependent on the primary keys.

**Solver** A Microsoft Excel add-in that is a software program developed for extending the capabilities of larger programs. Solver add-in consists of following files: Solver.xla, Solver32.dll, and Solvsamp.xls. These files calculate solutions to what-if scenarios based on adjustable cells, constraint cells, or cells that must be either maximized or minimized. Solver changes the decision variable to move from one feasible solution to another until the objective function has reached its optimal value.

**Structured Query Language (SQL)** An abbreviation of structured query language that is a standardized query language for requesting information from a database. Historically, SQL has been the favorite query language for database management systems running on minicomputers and mainframes. Increasingly, however, SQL is being supported by PC database systems because it supports distributed databases. This enables several users on a local-area network to access the same database simultaneously.

**Third Normal Form** The third state of tables in the normalization in the table design. In the Third Normal Form, no non-primary key attribute is transitively dependent on the primary key.

**Update Anomalies** The problem resulted from redundant data in the relations. They can be classified as insertion anomaly, deletion anomaly, and modification anomaly.

**Visual Basic (VB)** A programming language and environment developed by Microsoft. Based on the BASIC language, Visual Basic was one of the first products to provide a graphical programming environment and a paint metaphor for developing user interfaces. The Visual Basic programmer can add a substantial amount of code simply by dragging and dropping controls, such as buttons and dialog boxes, and then defining their appearance and behavior.

**Visual Basic for Applications (VBA)** A powerful language and development environment built into the Microsoft Office family of applications, which provide developer with professional quality development tools for building custom solutions. VBA comprises a VBA engine and an integrated development environment (IDE) with a full-featured editor, debugger, and OLE browser. It supplies basic control structures, math and string functions, and variable manipulation capabilities that enable developers to learn a single language and development environment which can then be used across multiple applications.

## VITA

Li Wang

Candidate for the Degree of

Master of Science

Thesis: OPTIMAL FERTILIZER INVESTMENT SYSTEM A COM – BASED  
APPLICATION

Major Field: Computer Science

## Biographical:

Education: Received Bachelor of Arts degree in Information Science from Beijing Polytechnic University in China, in July 1983; Attended Southern Polytechnic State University, Marietta, Georgia, from January 1997 – May 1998; Completed the requirements for the Master of Science degree in Computer Science at Oklahoma State University in December 2001.

Professional Experience: Teaching Assistant in Computer Science Department, Oklahoma State University, from August 2000 to May 20001. Column Editor for the journal, *Study of Political Science*, in the Institute of Political Science of Chinese Academy of Social Sciences, from July 1983 to September 1990.