REENGINEERING OF

COURSE ADVANTAGE

An Honors Thesis (HONRS 499)

by

Rodney A. Gouker

Thesis Advisor

Dr. Wayne Zage

Ball State University

Muncie, Indiana

May 1995

Graduation: May 1995
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Overview of <em>Visual Basic</em></td>
<td>3</td>
</tr>
<tr>
<td>Modifications of <em>Course Advantage</em></td>
<td>9</td>
</tr>
<tr>
<td>Improvements Made to <em>Course Advantage</em></td>
<td>19</td>
</tr>
<tr>
<td>Future of <em>Course Advantage</em></td>
<td>21</td>
</tr>
<tr>
<td>Appendix A - Software Life Cycle</td>
<td>22</td>
</tr>
<tr>
<td>Appendix B - <em>Visual Basic</em> Tool Box</td>
<td>23</td>
</tr>
<tr>
<td>Appendix C - <em>Visual Basic</em> Form Properties</td>
<td>24</td>
</tr>
<tr>
<td>Works Cited / Bibliography</td>
<td>25</td>
</tr>
</tbody>
</table>
INTRODUCTION

The concept of Course Advantage was introduced by Robert Brinkley and Gary Pavlechko, Ball State University Instructional Designers. Their idea was that Course Advantage would be a tool that teachers could use to create and edit lesson plans and course designs. This tool would facilitate dividing their course into Units, Lessons, and Modules. Each of these parts can then be named and contain detailed plans of the course components. The original program, Course Advantage, was developed by Dolores Zage in the C programming language. This DOS program used a series of note cards to graphically display the course material.

As computer technology advanced, changes became necessary and Course Advantage needed to be updated. A group of software engineering students at Ball State University was given the task of rewriting the program for the Windows environment. They completed this product in the spring of 1994. They used the programming language Visual Basic, a programming language for Windows that was just becoming popular. Visual Basic is a very powerful language that will be discussed in more detail later. The program then became a copyrighted product of Ball State University. Any problems with the program were directed back to Wayne Zage, the professor of the software engineering course sequence. In the fall of 1994, Course Advantage needed some more modifications. These modifications, which are discussed later, will allow the program to be distributed by Ball State University as a commercial product.

My search for an honors thesis topic began by choosing an advisor. I choose Dr. Wayne Zage, my software engineering professor, and we discussed possible thesis topics. Initially, I wanted my thesis topic to relate to my internship with Electronic Data Systems
Initially, I wanted my thesis topic to relate to my internship with Electronic Data Systems during the summers of 1992, 1993, and 1994. However, Dr. Zage suggested that I join the Ball State University Design Metrics Research Team and relate my thesis to the group’s efforts and research. I agreed to attend a meeting, evaluate my possibilities, and reach a decision about joining the group.

When I became part of the research group, under the direction of Dr. Zage, I was assigned to the Course Advantage project. I had some experience with Visual Basic from my internships with Electronic Data Systems and became the leader of a small group making the modifications to the program. "Reengineering of Course Advantage" refers to these modifications.

In software engineering, reengineering is defined as the renovation of an existing system (Galler and Corns, 395). These renovations may be made at any phase of the software development life cycle. The software development life cycle consists of eight steps as shown in Figure 1 (Pfleeger, 408). (Each step of the Software Life Cycle is discussed in detail in Appendix A.)

The renovations of Course Advantage involve several parts of this development process. Each renovation must be treated separately to ensure a correct and accurate program that satisfies its requirements.

![FIGURE 1: THE SOFTWARE DEVELOPMENT LIFE CYCLE]
OVERVIEW OF VISUAL BASIC

Visual Basic for Windows was first introduced by Microsoft in 1992 (Harris, 1). Since that time Visual Basic has become increasingly more popular. According to the Microsoft Visual Basic software Programmer's Guide, it has become popular because it is the quickest and easiest way to create applications for the Windows operating system (1).

Visual Basic is different from traditional programming languages because it is not just a language. It is a software framework that is basically graphical in nature. C++, on the other hand, uses a software framework called a class structure and is actual code. Visual Basic does not use any code in its software framework and thus the developer starts actual coding with part of his program already done.

The beginning of development for a Visual Basic program starts with the screens, or forms as called in Visual Basic. In other words, the first step is creating an interface that the user will see and use (Microsoft Programmer's Guide, 31). This interface is created using a series of forms, controls, and a minimal amount of coding.

The controls are used to get and receive information from the user and are placed directly on the form. There are several types of controls (See Appendix B). Each control has several properties and events associated with it. The properties are used to customize the control so that it looks a specific way. The events are used to perform a task when a specific operation has been done to a control. Events cause the control to act a specific way.

Forms also have properties and events associated with them. These properties and events are similar to a control’s properties and have the same purpose, namely, for
Forms also have properties and events associated with them. These properties and events are similar to a control's properties and have the same purpose, namely, for the form to look and act a specific way. Appendix C shows a picture of a form's properties that was copied from a blank Microsoft Visual Basic form. Using all of the properties and events from controls and forms, a computer program in Visual Basic can be started extremely quickly without writing a single line of code.

To illustrate an event occurring, suppose that the user clicks on a control. This causes its event click code to be executed. By putting code to open another form in this procedure, event click, the interface can become a prototype. Forms can thus be linked together with code to open and close forms, creating a prototype.

A prototype is extremely useful in software development. A prototype allows the client-partner, or customer, to see the program before it's completed. This ensures that the developer is correct with his requirements and that the screens are what the client-partner wants. When I worked for Electronic Data Systems, one project I worked on dealt with a prototype to show the client-partner. This prototype tried to persuade the client-partner that a computer program could be used to automate a specific process. This prototype in the end helped to secure the computer project.

After a prototype has been shown to the client-partner and satisfies the requirements, coding behind the forms begins. This coding involves the details of the program and how the program will function. This may include the development of a database, the generation of a sorting routine, or perhaps the loading and saving of a file. Each of these tasks in Visual Basic is easier than in C or C++. 
Rodney A. Gouker

The development of a database is easier because of a utility in Visual Basic called ODBC, Open Database Connectivity. This allows for database files to be accessed and changed easily. The database files are usually developed in a database application such as Microsoft Access and then used by Visual Basic. Visual Basic can perform record changes, record additions, or record deletions on the database file. Visual Basic knows the exact form of the database file based upon the ODBC driver being used. There are several drivers corresponding to several different database packages. The advantage of using the ODBC driver is that a database file created by Access can be used by Visual Basic or Access. Thus the developer does not have to create his own database file. According to Steve Guthrie, a programmer for Meridian Insurance, this is the most powerful part of Visual Basic and helped make Visual Basic popular.

The generation of a sorting routine is quicker because of the event coding. Each procedure of the event code is listed and can be accessed easily and quickly. This allows for a sort routine to be developed quickly by the user. While the actual code of the sort routine will be similar to the PASCAL or BASIC programming language’s code, it is still quicker for the developer to write the routine in Visual Basic. This is due to the data accessing mentioned in the previous paragraph. The ODBC driver contains several functions and procedures that allow for quick data access.

The loading and saving of a file is easier because Visual Basic is Windows oriented. By using the Windows environment, Visual Basic was able to become popular as Windows became popular. This allowed for Visual Basic to establish and use Windows standards. One standard which Visual Basic uses is the use of Common Dialog
Boxes. There are Open, Save As, Print, Color, and Font Dialog Boxes. These boxes each perform a specific task. For example an Open File Common Dialog box allows the user to select a file using the standard open file window. After these Common Dialog Boxes have been used, the developer must code the actual procedures to load the filename specified in the Dialog box. This is one type of *Windows* standard that *Visual Basic* adopted to allow easy operation for users. Other *Windows* standards include the formats for the menu design, the screen design, and the use of the program in general.

In *Visual Basic*, the design of a menu is quick and easy. Menu items are created for a form by selecting the form, and choosing Menu Design from the window menu item. Figure 2 is then displayed and can be used to enter the selected form's menu items. The caption is the name displayed for the menu item. The name is used to relate the menu items to their own events. Upon completion of the menu design, an outline of the menu items will be shown at the bottom of this window. Using this procedure in the designing of menu items allows for the menu items to become standard. For example, a File menu item should be first, followed by an Edit menu item, etc. These conventions or standards allow a user to easily find a menu item in any *Windows* application.
Visual Basic also implements the Windows standard for the help file. A help file can be created in Visual Basic as a companion to the program. The help file will use the Windows help system to display the developer’s help file. Even though the help system is extremely complex and requires a lot of time to learn, it is a good tool to use in Windows applications.

Visual Basic implements its programs to work like Windows programs. Since the developer is writing a program to run in the Windows environment, it makes sense for the program to operate like other Windows applications. However, since Windows has become popular and documentation on Windows is easy to find, a program written in Visual Basic can be learned quickly and easily. Almost everyone knows how to use Windows, so almost everyone should be able to use an application written in Visual Basic.

Visual Basic creates applications quickly and easily. It also makes its applications easy to use and understand. According to Steve Guthrie, the following are the strengths of Visual Basic:

1) It is the only package needed because it contains an editor, compiler, debugger, and report writer.

2) It is strongly supported by Microsoft and other documentation. Also over 1,000,000 copies have been sold.

3) There are not any runtime royalties.

4) It allows for quick prototyping with little code being written.
4) It allows for quick prototyping with little code being written.

5) It creates Windows applications — Windows is the platform most widely used.

6) It allows for object oriented programming.

7) It uses event driven programming.

These strengths along with others that I mentioned allow for Visual Basic to be the most efficient and powerful programming language developed. Because Windows is increasing in popularity, Visual Basic is not only a language of the present, it’s a language of the future.
MODIFICATIONS OF COURSE ADVANTAGE

Reengineering of Course Advantage began with the final goal being a product ready for distribution. The modifications identified covered several areas of the software development life cycle. Most of the modifications dealt with minor errors in the program and the documentation. The following errors and additions to the program were identified in the beginning of the project:

1) Grammatical Errors in the Text Boxes, Help Screens, and User’s Guide need to be removed.
2) “About” Box exited from program when it was supposed to simply close.
3) Menu Control Box on “About” form did not exit when double clicked.
4) Courses saved as untitled seemed to disappear.
   (No Extension)
5) Dragging causes stack overflow. Program Ends.
6) Installation into C:\CA directory overwrites previous versions.
7) Course name is not displayed at top of the screen when the Course box is clicked on.
8) Help does not follow Windows standards.
9) Menu items do not follow Windows standards.
10) Pasting text from the clipboard that contains Double quotation marks causes an error.
10) Pasting text from the clipboard that contains Double quotation marks causes an error.

11) Cut and Paste need to be stressed more in the Documentation.

These items needed to be fixed before the program could be distributed. A wish list of items that would be good features, but do not have to be completed are the following:

1) Ability to put Clip Art in Description boxes.

2) Increasing the size of the Course Map. It is currently limited to the size of the screen.

3) Dragging a part of the Course to a different location.

4) Opening more than one description box at a time.

Currently, Only one can be opened at a time.

The next several pages will discuss all of the items mentioned above and how they were corrected.
1) **Grammatical Errors in the Text Boxes, Help Screens, and User’s Guide need to be removed.**

8) **Help does not follow Windows standards.**

11) **Cut and Paste need to be stressed more in the Documentation.**

   These errors were relatively easy to fix. The errors in the text boxes were simply corrected. As for the help screens and the user’s guide, they are both being rewritten. The help screens are being rewritten to follow standard Windows format and to contain new information on the changes made to *Course Advantage*. The User’s Guide is being rewritten to provide information on the new parts of *Course Advantage* and to have better documentation for the program. It is also being rewritten to stress the cut and paste options more.

2) **“About” box exited from program when it was supposed to simply close.**

3) **Menu Control Box on “About” form did not exit when double clicked.**

   These errors required some coding in the “About” form events. The changes were relatively minor. Involved commenting out a line of code that reloaded the form.

4) **Courses saved as untitled seemed to disappear. (No Extension)**

   This error also required some coding in the “Mod_Box” form events. The changes involved checking if the course was untitled and calling the Save_As Procedure. Also, to prevent the loss of a course upon exit, the same code was used in the unload procedure.
5) Dragging causes stack overflow. Program Ends.

Wish List Item: 1) Dragging a part of the Course to a different location.

The problem with the stack overflow has only been discovered on a few computers and does not occur every time. It is believed that dragging is a very difficult process. We determined that dragging was going to take more time than it is worth. Thus, our solution is to improve the cut and paste so that the subordinates of a part of a course can be cut and pasted. This improvement has not been completed yet, but should be done by June 1.

6) Installation into C:\CA directory overwrites previous versions.

We determined that this was not a major problem, because anyone using the Windows version would not want the DOS version anyway. Also, the installation screens allow the user to change the directory into which the files are being installed. It was, however, an idea to have the installation check for the previous version and prompt for an overwrite. This improvement has also not been completed and it is undetermined whether or not it is possible.
7) **Course name is not displayed at top of the screen when the Course box is clicked.**

This problem required the coding of displaying the course in the appropriate event procedure. The event was click on the course box. The code simply displayed the course name where the unit, lesson, and module names are placed when they are selected.

9) **Menu items do not follow Windows standards.**

The Menu items were redone using the *Windows* standard. The process for changing a menu item was discussed previously and seems trivial at this point.

10) **Pasting text from the clipboard that contains Double quotation marks causes an error.**

This was a major problem with the program. *Visual Basic* interrupts double quotes as some type of marker. We pasted some text and received an error message generated by some coding by the software engineering students. This error message was commented out and code was put in its place to check for double quotes and change them to single quotes. This procedure was also used when a double quote was typed from the keyboard. This coding was done in each place that a paste occurred and the error message was generated.
Wish List Item: 1) Ability to put Clip Art in Description boxes.

This item was determined to be very difficult to implement. It was determined that the method for doing this would be to change the text boxes to another control that accepts graphics and text. We agreed that a simple comment about a picture could be used instead of a clip art.
Wish List Item: 2) Increasing the size of the Course Map.

This was a major change that was implemented. Changing the size of the screen involved several smaller steps. First, determine how the course map was currently being printed. Second, determine a plan on how to implement the imploding and exploding units. Third, change the number of units, lessons, and modules to a global constant. Making this change required a lot of testing. It was extremely difficult to find the location of every reference to the size of one of the items. Fourth, change the course map print procedure to print only the selected unit. Fifth, add an array to any units that contained subordinates and was not the selected unit. When this was completed, the imploding and exploding screen looked as shown in Figure 3.

![Course Map Screen](image-url)
Wish List Item: 4) Opening more than one description box at a time.

This was also a major change that was/is being implemented. This item may also be broken into several parts. First, we changed the properties of all the forms. Second, we created a MDI parent form that is used to keep all of the windows inside. It was first thought that the course map would be the MDI parent. However, MDI parent forms cannot contain control except for buttons. Thus the course map and all of the other forms are child forms of the MDI form. Third, a strategy was implemented to cause the course map to be maximized and the other forms not to be. Fourth, we added menu items to list the current windows. This is where the program is currently. We plan to finish by doing the following steps: Fifth, write code to tile and cascade the windows with the course map minimized. Sixth, write code to allow the same form to be opened many times with different data in each one. These items should be completed in a couple of weeks.
Several other problems were found with the program as we worked with it, and these problems also needed to be fixed before the program could be distributed.

“New” course caused existing course to be lost.

“Open” course caused existing course to be lost.

These problems involved creating a message box, prompting to see if the current course needed to be saved, and calling save if necessary to save the changes. The code was copied into two procedures. It was used in the menu item “new” click event and in the menu item “open” click event.

Several related problems with Common Dialog boxes and filenames.

These problems involved invalid filenames and changes with the Common Dialog boxes. One problem was the length of the filename. Changing the Common Dialog box flags allowed for the filename to be checked easily. This problem was evident in several locations and thus was changed in all locations. There was also a major problem with the menu control box on the course map form and the saving of the course when exiting using this method. The control box was fixed so that the exit choice and the control box performed the same procedure.
Problem related to Saving and Loading the Course.

This problem has not been fixed yet. We believe that the problem is in the open procedure and that it does not assign a value to a variable. This value corresponds to the value to use for the next subordinate. The problem is after loading the file and creating a new subordinate, the new subordinate and the first created subordinate are somehow linked together. That is, when the name of one is changed, the other changes also. This problem was believed to be created by the Course Map size Fix. However, after further study it was determined that it was an original flaw in the program.
IMPROVEMENTS MADE TO COURSE ADVANTAGE

The original Course Advantage written in C met all of the requirements at that time. However, the program written by the software engineering students had several new requirements. These new requirements caused them to use Visual Basic and develop a Windows application. The modifications, made to Course Advantage and listed in the previous section, have made it better.

The second version of Course Advantage in Visual Basic is better because it meets several new requirements. One new requirement would be the ability to have a larger course size. The first version was limited to the screen. The second version allows for courses to have up to 20 units with each unit having up to 12 lessons. Each lesson can have up to 15 modules. This allows for an extremely large course to be entered into the revised Course Advantage. In general, the new requirements are the errors and wish list items mentioned earlier. These errors and items each allow for the system to better.

Another reason for the second version being better is the design of the program. While we were not able to correct all of the design errors, several were corrected. One example would be changing the index sizes to global variables to allow for easy modification at a later date. Another example would be causing the program to run the same code when a control box is double clicked or when the exit option is chosen from the menu. These design changes make the system run more efficiently.

One item we were not able to improve was the code. Currently, the entire course is stored in a global array without any parameters ever being passed between procedures. In the future, I would hope that the system would be revised to contain only the necessary
In the future, I would hope that the system would be revised to contain only the necessary global variables. Another flaw in the existing code is the lack of using the option explicit coding strategy in Visual Basic. These means that variables in the current system do not have to be declared. If they are not declared the variables are assumed to be of type variant. Variant data type variables do not require any extra storage. They are, however, a bad programming practice. It is much easier to determine what a piece of code does if the variable declarations are present. We plan to implement the option explicit strategy. However, this will take a large amount of time.

The Visual Basic version of Course Advantage is easier to test than the C version. This is due to the lines of code required in each language. There is, however, more testing required in the second version of Course Advantage in Visual Basic. This is due to the new requirements mentioned above. The same tests from version one must be used, along with new test cases for the new requirements and functionality.

Since the Visual Basic version has fewer lines of code than the C version, the Visual Basic version has fewer errors and is easier to maintain. The second Visual Basic version has even fewer errors and thus should be even easier to maintain. Along with new and accurate documentation for the user, new programming documentation will be developed. This new documentation will also help in reducing the maintenance time.

Overall, the second Visual Basic version of Course Advantage is much improved compared to previous versions. The new functionality and new documentation help to provide a very effective tool for teachers to use. Course Advantage is the course development tool of the future.
FUTURE OF COURSE ADVANTAGE

Course Advantage, after June 1, will undergo intensive testing. This testing is used to ensure that the program is ready for distribution. I perceive that Course Advantage will pass the testing with only minor flaws. These minor flaws will be fixed quickly and easily and Course Advantage will be ready for distribution. I see Course Advantage having a difficult time being sold at first. However after about one year's time, I see Course Advantage becoming a widely used tool. I see students using it at universities for developing lesson plans. I see teachers using it in the development of their courses. I see Course Advantage as being a valuable tool in the future.
Requirements Analysis -- What is the problem? In this stage the developer researches the current situation and documents the details of the problem. This stage includes the development of the requirements document.

System Design -- What is the solution? In this stage the developer evaluates possible solutions and documents generalizations of the solution chosen. This stage includes the development of the Data Flow Diagram.

Program Design -- What are the mechanisms that best implement the solution? In this stage the developer documents the solution in great detail. This stage includes the development of a structure chart and pseudocode of the proposed solution.

Program Implementation -- How is the solution constructed? In this stage the developer actually writes the code detailed by the design.

Unit and Integration Testing -- Is the implementation working as designed? In this stage the developer tests the system based on the units designed earlier.

System Testing -- Is the problem solved? In this stage the developer ensures that the system follows the requirements documented earlier and performs correctly.

Delivery -- Can the customer use the solution? In this stage the developer gives the system to the customer. A tutorial may need to be given so that the customer can use the system without the developer.

Maintenance -- Are enhancements needed? In this stage changes or enhancements are made to the original system. This stage is never completed for a given software product. The product will always require enhancements as technology changes.
The following is a picture of the tool box from *Visual Basic* 3.0 Professional.

Each control’s name is listed starting with the first row and then continuing through the rows to the bottom.

1) Pointer
2) Picture Box
3) Label
4) Text Box
5) Frame
6) Command Button
7) Check Box
8) Option Button
9) Combo Box
10) List Box
11) Horizontal Scroll Bar
12) Vertical Scroll Bar
13) Timer
14) Drive List Box
15) Directory List Box
16) File List Box
17) Shape
18) Line
19) Image
20) Data
21) Grid
22) OLE
23) Animated Button
24) Common Dialog
Note - Look at all of the properties. Imagine all the different settings that are possible for this one form. Each control has approximately the same number of properties, but may also contain some new properties in place of some other properties.
WORKS CITED / BIBLIOGRAPHY


