Developing a Web-Based Engineering Economy Courseware

Chan S. Park, Jai W. Kang

Auburn University/ Rochester Institute of Technology

Introduction

Recently, more web-based instructional materials have been developed to help students in learning various engineering subject matters. In engineering economy instruction in particular, the use of the Internet is increasingly common. Even though most instructors as well as students appreciate the power of the Internet, how we develop a web-based courseware is a much debated issue. In particular, Excel becomes almost the de facto computing platform in solving various engineering economic decision problems. In this presentation, we demonstrate the general scheme of developing web-based engineering economy courseware, taking advantage of the Internet as well as the well-known Excel computing platform.

In this paper, we are proposing a new way of developing engineering economics course materials based on the Web. In doing so, we are developing a Java based on-line processor and linking to an Excel spreadsheet for more complex economic analysis. All major engineering economics problems will be classified into one of two categories: Category 1—simple calculation, which can be accomplished by Java-Calculators; Category 2—requires extensive tabular presentation, suitable for Excel spreadsheet. All user interfaces will be done on the web, so that students can respond to basic input queries. Then, the program will create an Excel file ready for economic analysis. The entire process is somehow similar to using Turbo Tax on a PC—you answer the basic questions, then Turbo Tax creates the suitable tax return forms with blanks filled. Then, the users can interactively change the data directly on the screen to conduct "what if" analysis. A comprehensive replacement analysis will be demonstrated in the paper.

Developing On-Line Financial Calculators

The first category is to develop an integrated financial calculator based on Java, so that the calculator can be accessed with any Web browser. In either classroom instruction or students working on homework, it is often desirable to perform simple operations such as finding net

present values (NPV), rate of return, or other equivalence calculations. Certainly, it is possible to do all these computations on Excel. However, if the main purpose is to obtain the result for quick decision, it is much easier to do these using on-line financial calculators. For example, with Excel, to obtain the internal rate of return for a series of investment cash flows, the user has to input those into Excel cells and call for the IRR function command to compute the value. In doing so, the user has to understand the exact format to input the required data entries for the command. To produce some graphical output such as an NPV plot (NPV as a function of interest rate), it will take several steps to obtain the desired result. However, with a well designed on-line financial calculator, we just input the basic cash flows and press the compute key to see the result as well as the graphical output. The key difference is that the user does not need to know the computational steps or command sequences to go with the process.

Cash Flow Analyzer (Java Program)

Cash Flow Analyzer (CFA) as shown in Figure 1 is an integrated cash flow analysis program written in Java, which includes various computational modules such as present value, future value, annual worth, benefit cost ratio, payback period, discounted payback period, internal rate of return. It also allows the user to obtain a graphical output for present value curve, project balances, and cash flow diagram. It has a cash flow input data editor that automates the data entry process. We will briefly discuss the design principles adopted for CFA.

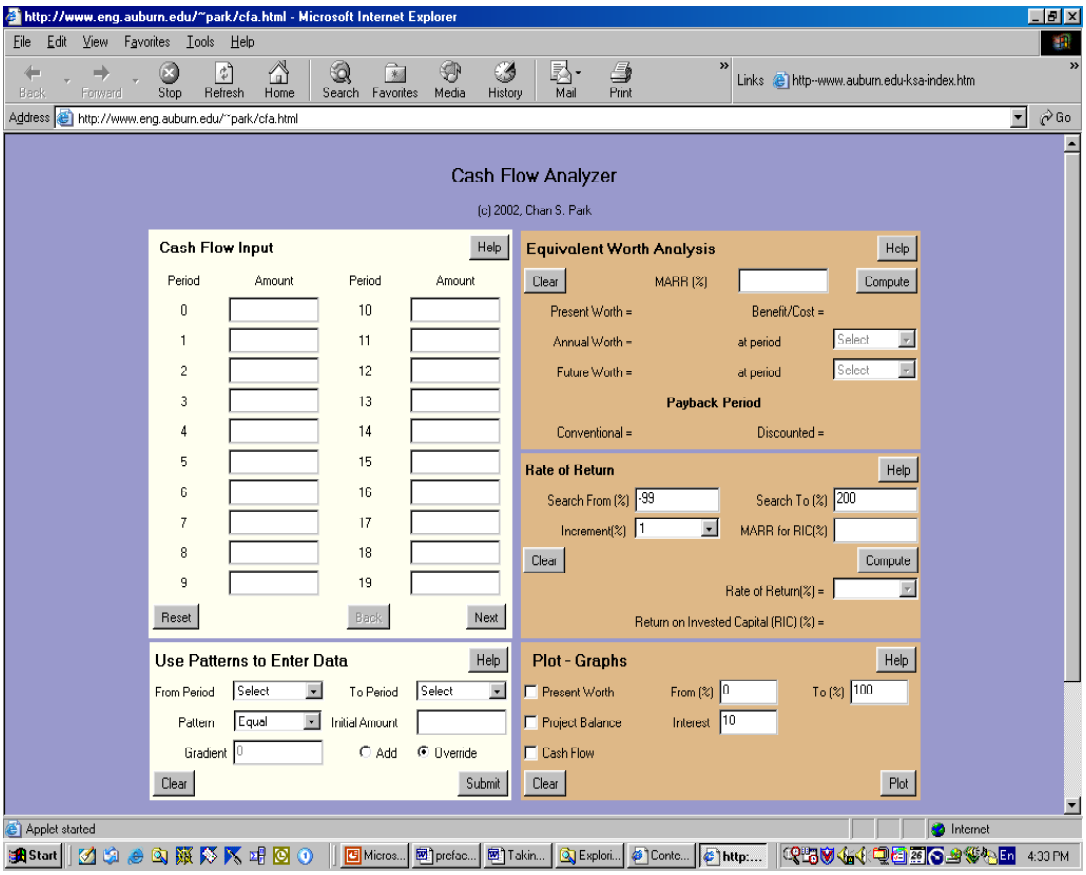


Figure 1 Cash Flow Analyzer – Menu Screen

Design Principles

A first step toward any on-line financial calculator is to provide a mechanism for furnishing data to the program. For this purpose, we commonly develop a general-purpose data editor program for creating an input data file and editing it when necessary. Good editing features are critical to the financial calculator. These features make it possible to avoid restarting or backing up a long data entry sequence when the operator errs in an entry or wishes to modify the data once it has been completely loaded into memory. We will first describe some of the design principles of Cash Flow Data Editor. These design concepts, based on human engineering principles, are applicable to any on-line financial calculator.

Feedback

Feedback is an important part of an interactive system. Typically, feedback occurs as each command is accepted by the system. For example, a command picked from a menu is highlighted so that the user knows that the action has been accepted. The most useful form of feedback also tells the user that the requested operation has been completed. This is usually done with a new or modified display that explicitly shows

the results. Audio feedback such as a beep sound has the same effect. If a requested task will take sometime to complete, the program should provide another type of feedback to indicate that the computer is at least working on the problem. The Cash Flow Data Editor program uses several such feedback mechanisms.

Consistency

The position on the screen of items such as menu, commands, and feedback is important; we normally designate a fixed portion of the screen for each of them. This consistency allows the user to build "muscle memory" that helps in picking the desired menu item quickly. Another example of consistency is in the use of color coding. Colors should always convey information in the same way (e.g., a red character means change in input data). Another important principle is to make necessary memorization as simple as possible and to avoid it whenever possible.

Description of the Cash Flow Data Editor

As shown in Figure 1, the left-hand side of the screen is designated as the area of input data. The data editor has the following features:

- The user can enter cash flow series up to 400 periods, which is sufficient in most economic analysis.
- With the "overwrite" mode, the user can enter the cash flow data by overwriting the data already in the cell. With the "add" mode, the data entered will be added onto the data already in the cell.
- The user can automate the input process for cash flows with known patterns, such as uniform, linear or geometric gradient series, by selecting the "Use Pattern to Enter Data."

- Unless the user reset the data, all subsequent computations will be based on the data stored in the current cells.

Description of Equivalent Worth Analysis

First output area (upper right-hand side) is to display the numerical values of various measures of investment worth. In doing so, it requires to enter the required rate of return (or minimum attractive rate of return. Once the MARR is specified, the following figures of merit will be obtained:

- Net present value (NPV)
- Annual worth (over *n* period as selected by the user)
- Future worth (at *n* period as specified by the user)
- Payback period
- Discounted payback period
- Benefit-cost ratio

Second output area is to display the rate of return figures of the project. The method used to find the rate of return is based on the search method, so that the user must specify the range as well as increment of the search. In particular, the user can find the following in this display area:

- Rate of return
- Internal rate of return
- Return on invested capital (for a mixed investment)

As expected, the rate of return is defined as the interest rate that makes the present value of inflows equal to the present value of the outflows. Naturally, there is the possibility of multiple rates of return for any mixed project. If there are, the CFA will find all of them within the search limit. One way to make sure that you find all the rates of return for a given mixed investment, you need to resize the upper search limit to include all the rates of return possible. Initially, the search range is preset between –99% and 200%, with a search increment at 1%. Once you locate the rate of return at this initial trial, you can refine the search with a smaller range with a smaller increment.

If you are dealing with a mixed investment, the rate(s) of return found may not serve as the true figure of merit. To find the true rate of return [or commonly known as the return on invested capital (RIC)], you need to specify the external interest rate (or MARR, the interest rate used in computing the NPV). The RIC will be calculated as a function of this external interest rate specified[5] . If you are dealing with a conventional project or a pure investment, the rate of return and RIC should be the same. If Excel is used to find this RIC, it would take some extra modeling effort. However, with CFA it is a matter of a few key strokes to determine the true rate of return figure.

Plot-Graph Area

The third area for output display is reserved for various plots. Basically, there are three plots that can be obtained for a given cash flow series.

- Present worth plot
- Project balance plot
- Cash flow plot

You can obtain the present worth plot as a function of interest rate (or MARR). Initially the range is set at between 0% and 100%. However, the user can change the range as desired. The project balance plot as a function of project period is also obtained by specifying the interest rate. Finally, the cash flow plot is easily obtained as well. Each plot is displayed in a new window, so that the user can make various "what if" comparisons by changing the interest rate or altering the original cash flow data.

Example 1

Consider the following cash flow given below:

Year	Cash Flow
0	-$200
1	150
2	50
3	50
4	50
5	50
6	50
7	50
8	-285

(1) At MARR of 6%, find the net present value.
(2) Compute the rate(s) of return. If this is a mixed investment, find the RIC at 6%.
(3) Plot the NPV as a function of interest rate (0% - 100%)

Solution:

First enter the required cash flow series as shown in Figure 3. At MARR of 6%, the net present value is -$5.35, indicating that it is not a good investment. The project has multiple rates of return at 8.74% and 17.46%. Since this is a mixed investment, it is necessary to find the true internal rate of return (or return on invested capital). At MARR of 6%, the RIC is computed at 3.84%. Since the RIC is less than MARR, the project is not acceptable. The NPV plot is also found in Figure 4. It clearly shows that there are two rates of return, indicating this is a mixed investment. (If any of the project balances calculated at the project's rate of return is positive, the project is defined as a mixed investment.)

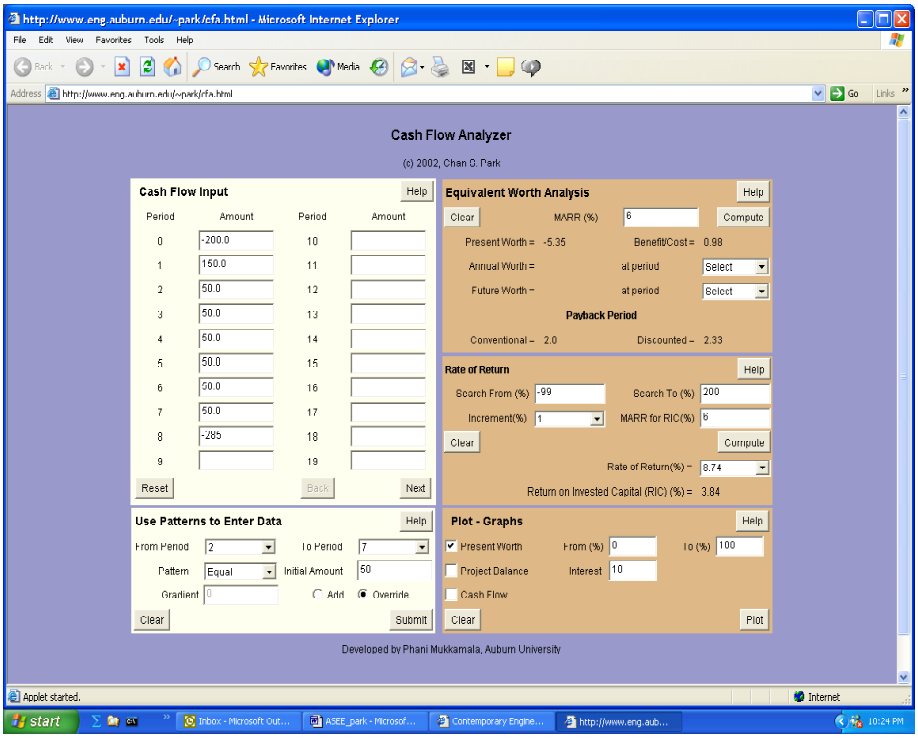


Figure 3 Output Display for Example 1

(Note: The CFA program can be accessed at http://www.eng.auburn.edu/~park/cfa.html.)

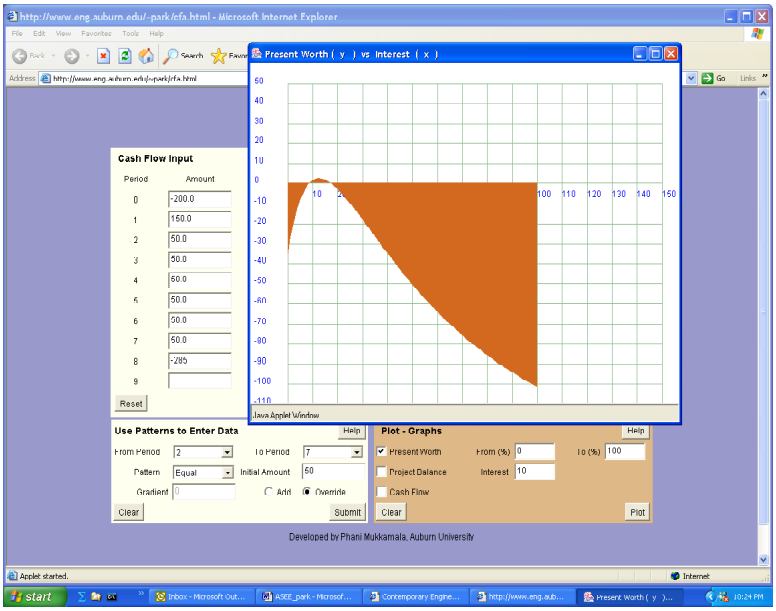


Figure 4 NPV plot as a function interest rate (Example 1)

Excel file creation for engineering economics problems requiring extensive tabular/graphical presentations

This section discusses the second category of developing engineering economics course materials based on the World Wide Web. When major engineering economics problems require extensive tabular and/or graphical presentation, it is suitable for students to utilize an Excel spreadsheet. We discuss here a Web-based application development. All user interfaces are performed on the Web, so that students can respond to basic input queries. Then, the program creates an Excel file, ready for further economic analysis such as the "what if" analysis as well as a resulting graphical presentation of the analysis.

Objective

The objective is to create Excel files by inputting problem initial data on the Web to analyze Engineering Economics problems requiring extensive tabular presentation.

Development Platform:

- MS Visual Basic 6 to create ActiveX Document DLL Project
- ActiveX EXE component: MS Excel 9.0 Object Library
- MS Excel 97 or later version

User Environment:

- PC with Windows Operating System

- MS Internet Explorer (IE 4.01 and later)
- MS Excel 97 or later version

Approach

We can use and/or create an Excel file by referencing to the Microsoft Excel 9.0 Object Library from within a Visual Basic (VB) program[6]. Once adding the reference to the Excel object, we can access its objects including the Excel Application object, Workbook object, Worksheet object, and Range object. Each Excel file is stored as a Workbook. Each Workbook contains a number of Worksheets. Range objects can be created to reference a cell or group of cells in a Worksheet[4,7] .

When we create a VB project of ActiveX Document type, it can be run from a Web browser[1] . The ActiveX document is a container, called a UserDocument object, on which we can place controls just as we would for a VB form.

After creating and compiling the VB application, we can use the Package and Deployment wizard to create an Internet package to distribute the application over the Internet. The distribution package includes a .vbd file and a cabinet (cab) file. The cab file is a compressed file for the component and support files that are needed so that it can be run from the Internet Explorer Web browser on the user's PC.

Even though Internet Explorer 4.0 and later versions should automatically download the ActiveX document on the user's PC through a link to the .vbd file from a browser, we experienced difficulties especially when trying this on a PC without the VB runtime and OLE automation installed. Thus, we developed a three-step procedure available on the Web server for users:

Step1: Installation of the VB runtime and OLE automation via a link to an MS Web site for download[1],

Step2: Registration of the ActiveX document server on the user's PC, and

Step3: Creation of an actual link to the .vbd file.

When a user's PC does not have the VB runtime component installed, the above first step will download and install the required component. The second step is to register the actual program on the user's PC. Finally the third step allows the user to run the Web-based engineering economics analysis program by clicking the link to the vbd file.

The following is a summary of steps to develop a Web based VB program to generate an Excel file, assuming the reader understands the Microsoft Visual Basic 6 including compiling and the Package and Deployment wizard[2,3] :
:

[1] www.microsoft.com/downloads/release.asp?releaseid=28337&area=top&ordinal=5

Steps

1. Make a reference to the Excel object.
   - Start the VB6 IDE with a new ActiveX Document DLL project.
   - Click the Project menu and select the References option.
   - Check the Microsoft Excel 9.0 Object Library in the Reference dialog box's list box.

2. Design static User Interface on a UserDocument object.
   - A UserDocument is similar to a form in a Design window of a VB Standard EXE project.

3. Create any dynamic User Interface on the UserDocument object
   - Set up any data entry user interface to be determined at runtime depending on initial user input data. For example, we need to add seven TextBox controls dynamically during runtime for the user to enter O&M costs when the user enters a number of 7 on the # Holding Periods TextBox control.

4. Create an Excel File (UserDocument_Initialize() event procedure) within the VB project.
   - Create a new Excel Workbook.
   - Add an Excel Sheet to the Excel Workbook.
   - Write source codes to read in user data and manipulate them to store the results in the Excel Sheet.

5. Compile the VB DLL project.
   - Select the Make project_name.DLL option from the File menu.

6. Package the VB DLL project using the Packaging and Deployment Wizard.
   - Create a dependency file, which contains dependency information needed by the ActiveX DLL component.
   - Create an Internet package.
   - Create a standard package, which will be used to register the program on the user's PC in case the above VBD does not work as it is supposed to.

7. Set up a Web server for PC Users with Excel to access the Web page from the Internet Explorer.
   - Create a Web site link for installation of the VB runtime and OLE automation via a link to an MS Web site for download (see footnote 1).
   - Prepare a self-extractable .exe file from the setup package made in the above step 6, which includes a cab file, setup.exe and setup.lst. Make a link available to the user to register the program on the user's PC.
   - Deploy the .vbd and cab files into a Web server and establish a link on a Web page to the .vbd file.

Example 2 Economic Service Life Calculation

To illustrate the approach to develop a Web-based program to create an Excel file, we followed an example[2] [5] to generate a two-part worksheet. The first one is the after-tax salvage value generation, and the second one is the economic service life calculation on after-tax basis. Once again, our objective is to generate an Excel table. The required input data set might be

- Investment
- Tax rate (%)
- Current Book value
- MARR (%)
- Holding period of the asset—(example 7 years)
- Expected annual O&M costs over the holding period.
- Annual depreciation amounts over the holding period
- Expected market (salvage) value of the asset over the holding period.

Steps

1. Make a reference to the Excel object.
   - Start the VB6 IDE with a new ActiveX Document DLL project.
   - Click the Project menu and select the References option.
   - Check the Microsoft Excel 9.0 Object Library in the Reference dialog box's list box.

2. Design a User Interface on the UserDocument object. (Figure 5)
   - A frame control whose caption is "Input Data" contains five TextBoxes for the user to enter Investment, Tax Rate, Book Value, MARR and a number of Holding Periods.
   - Set up two control arrays: lblHP(0) and txtOM(0) where the label control array is used for displaying the holding periods and the TextBox control array will be used for the user to enter O&M, Depreciation and Expected Market Values. The sizes of both control arrays will be determined by the # Holding Periods that the user inputs.
   - A Command Button: **Set Up Data Entry Grids** is clicked to display placeholders for O&M, Depreciation and Market Values along with Holding Period labels once the user finishes entering the Input Data.
   - A Command Button: **Create Excel** will be used to create an Excel file.
   - A Command Button: **Reset** can be clicked to clear all the input data controls.

[2] http://www.eng.auburn.edu/~park/notes/notes_ch15.PDF

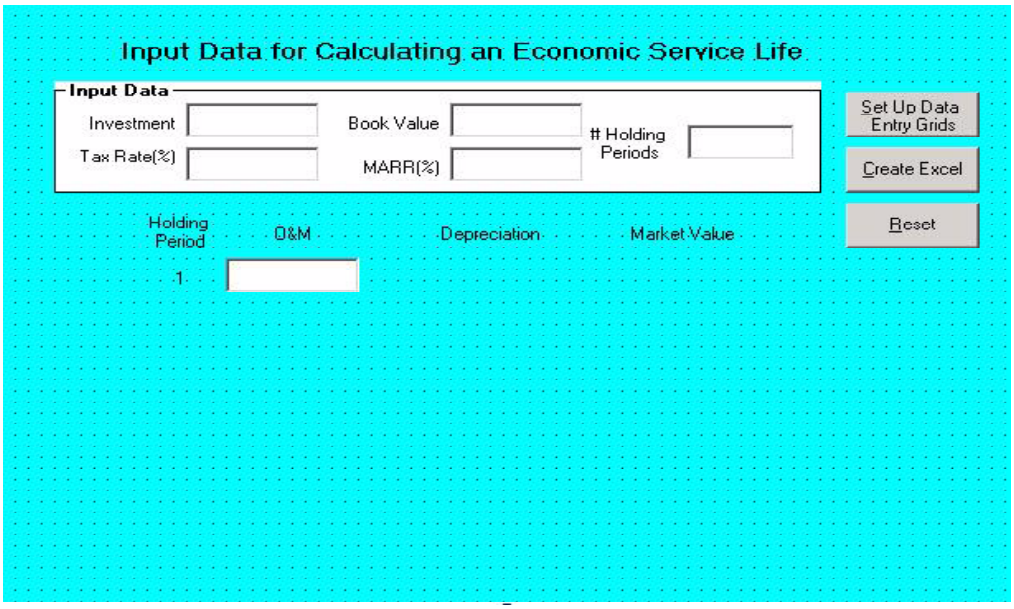
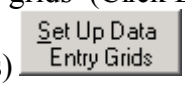


Figure 5 Initial User Interface

3. Set up data entry grids (Click Event procedure of command button: &Set Up Data Entry Grids) 

- Validate the input data: Investment ($), Tax rate (%), Book value ($), MARR (%), and Holding periods to be numerical using the IsNumeric() function.

- Set up Data Entry Grids for O&M, Depreciation, and Market Values depending on the holding periods that the user entered using control arrays.

- Display holding periods by dynamically loading a label control array (lblHP()) at runtime.

```
For intCount = 1 To mintPeriods - 1
    'Display Holding Periods Column
    Load lblHP(intCount)
    lblHP(intCount).Top = lblHP(intCount - 1).Top + 500
    lblHP(intCount).Caption = intCount + 1
    lblHP(intCount).Visible = True
```

- Create three columns and the same number of rows as the holding periods of TextBoxes for the user to enter O&M costs, Depreciation Amounts, and Expected Market Values by dynamically loading a TextBox control array (txtOM()) at runtime. For example, (Figure 6)

```
Load txtOM(intCount) 'Load TextBoxes for the O&M

txtOM(intCount).Top = txtOM(intCount - 1).Top + 500
txtOM(intCount).Text = ""
txtOM(intCount).Visible = True
```

Next intCount

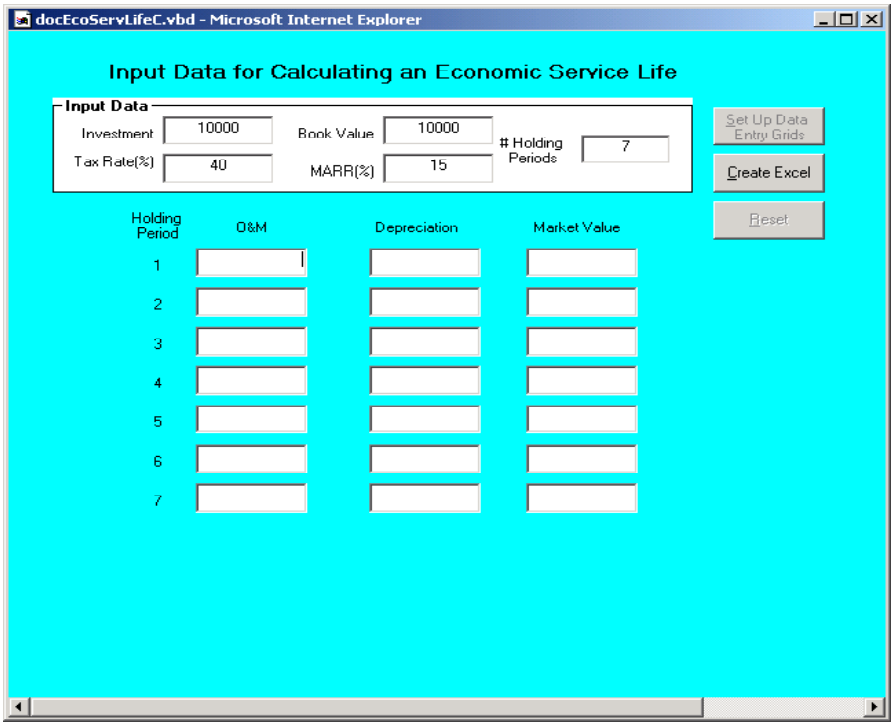


Figure 6 Data Entry Grids

4. Create an Excel Sheet of an Excel Workbook (Click Event procedure of command button: &Create Excel)

- Declare and create a new Excel Workbook.

```
Dim xlCurrent As Excel.Application  'declare a new Excel Workbook
Dim wbCurrent As Excel.Workbook

Set xlCurrent = New Excel.Application 'create a new Excel Workbook
Set wbCurrent = xlCurrent.Workbooks.Add
```

- Read in the input data: Investment ($), Tax rate (%), Book value ($), MARR (%), and the number of Holding periods. (Figure 7)

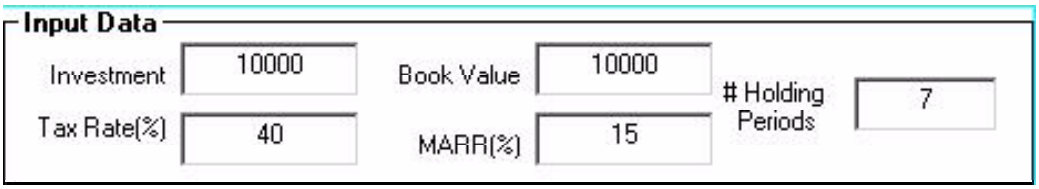


Figure 7 Initial problem definition data set

- Find the period of full depreciation from the user's input data (Figure 8), and fill in the Excel cells for the permitted annual depreciation amounts over the holding period (Figure 9).

Figure 8. User input of depreciation values

		Permitted Annual Depreciation Amounts over the						
Holding		*Holding Period*						
Period	O&M	*1*	*2*	*3*	*4*	*5*	*6*	*7*
1	$ 2,000	*$ 2,000*						
2	$ 3,000	*$ 2,000*	*$ 1,600*					
3	$ 4,000	*$ 2,000*	*$ 3,200*	*$ 960*				
4	$ 5,000	*$ 2,000*	*$ 3,200*	*$ 1,920*	*$ 576*			
5	$ 6,000	*$ 2,000*	*$ 3,200*	*$ 1,920*	*$ 1,152*	*$ 576*		
6	$ 7,000	*$ 2,000*	*$ 3,200*	*$ 1,920*	*$ 1,152*	*$ 1,152*	*$ 576*	
7	$ 8,000	*$ 2,000*	*$ 3,200*	*$ 1,920*	*$ 1,152*	*$ 1,152*	*$ 576*	*$ 0*

Figure 9. Excel cells for the permitted annual depreciation amounts

- Generate the after-tax salvage values. For example, the Net A/T Salvage Value column (Figure 10) is generated as follows:

```
Const TBL_A_BLANK_ROW As Integer = 10‘General Declaration section
Dim mintPeriods As Integer                    ‘ = 7 years in this example

Dim Rng, Rng1 As Object                              ‘cmdExcel_Click()
Dim strCurrFormat As String
strCurrFormat = "$* #,##0;$* (#,##0)"

With wbCurrent.Worksheets(1)
For k = 1 To mintPeriods
Set Rng = .Cells(TBL_A_BLANK_ROW + k, 5 + mintPeriods)
Set Rng1 = .Cells(TBL_A_BLANK_ROW + k, 7 + mintPeriods)
.Cells(TBL_A_BLANK_ROW + k, 8 + mintPeriods).Formula = "=" & Rng.Address &
    "-" & Rng1.Address
```

```
.Cells(TBL_A_BLANK_ROW + k, 8 + mintPeriods).NumberFormat = strCurrFormat
    Next k
End With
```

Holding Period	Total Depreciation	Book Value	Expected Market Value	Taxable Gains	Gains Tax	*Net A/T Salvage Value*
1	$ 2,000	$ 8,000	$ 6,000	$ (2,000)	$ (800)	***$ 6,800***
2	$ 3,600	$ 6,400	$ 5,100	$ (1,300)	$ (520)	***$ 5,620***
3	$ 6,160	$ 3,840	$ 4,335	$ 495	$ 198	***$ 4,137***
4	$ 7,696	$ 2,304	$ 3,685	$ 1,381	$ 552	***$ 3,133***
5	$ 8,848	$ 1,152	$ 3,132	$ 1,980	$ 792	***$ 2,340***
6	$ 10,000	$ 0	$ 2,662	$ 2,662	$ 1,065	***$ 1,597***
7	$ 10,000	$ 0	$ 2,263	$ 2,263	$ 905	***$ 1,358***

Figure 10 Net A/T Salvage Value

- Calculate the economic service life on the after-tax basis. For example, the total cost column (L) (Figure 11) can be generated using the AutoFill method as follows:

```
With wbCurrent.Worksheets(1)

 Set Rng = .Cells(TBL_A_BLANK_ROW + mintPeriods + 8, 10)          'J25
     Set Rng1 = .Cells(TBL_A_BLANK_ROW + mintPeriods + 8, 11)       'K25
     .Cells(TBL_A_BLANK_ROW + mintPeriods + 8, 12).Formula = "=" & _
        Rng.Address(False, False) & "+" & Rng1.Address(False, False)      'L25

      'Source for autofill
     Set Rng = .Cells(TBL_A_BLANK_ROW + mintPeriods + 8, 1)
     Set Rng1 = .Cells(TBL_A_BLANK_ROW + mintPeriods + 8, 12)
strRange = CStr(Rng.Address(False, False)) & ":" & CStr(Rng1.Address(False,
     False))

     'Destination for Autofill
     'Set Rng = .Cells(TBL_A_BLANK_ROW + 2 * mintPeriods + 7, 1)
     Set Rng1 = .Cells(TBL_A_BLANK_ROW + 2 * mintPeriods + 7, 12)
strRange1 = CStr(Rng.Address(False, False)) & ":" & CStr(Rng1.Address(False,
     False))

     .Range(strRange).AutoFill .Range(strRange1)
End With
```

5. Compile the VB project.
   - Click the File menu, and locate and select the "Make EcoServLife.dll" option.

6. Package the VB DLL project using the Packaging and Deployment Wizard.
   - Create a dependency file, which contains dependency information needed by the ActiveX DLL component.
   - Create an Internet package.

- Create a standard package, which will be used to register the program on the user's PC in case the above vbd does not work as it is supposed to.

Holding Period N	A/T Market Value	PV of Market Value	A/T O&M Cost	PV of O&M Cost	Cum PV of O&M Cost	Cum PV of Dep. Credit
1	$ 6,800	$ 5,913	$ 1,200	$ 1,043	$ 1,043	$ 696
2	$ 5,620	$ 4,250	$ 1,800	$ 1,361	$ 2,405	$ 1,180
3	$ 4,137	$ 2,720	$ 2,400	$ 1,578	$ 3,983	$ 1,916
4	$ 3,133	$ 1,791	$ 3,000	$ 1,715	$ 5,698	$ 2,300
5	$ 2,340	$ 1,163	$ 3,600	$ 1,790	$ 7,488	$ 2,547
6	$ 1,597	$ 691	$ 4,200	$ 1,816	$ 9,303	$ 2,761
7	$ 1,358	$ 510	$ 4,800	$ 1,804	$ 11,108	$ 2,761

	Equivalent Annual Cost if the Challenger is kept for N more Years		
Total PV of Operating Cost	Capital Cost	Operating Cost	***Total Cost***
$ 348	$ 4,700	$ 400	***$ 5,100***
$ 1,225	$ 3,537	$ 753	***$ 4,291***
$ 2,067	$ 3,188	$ 905	***$ 4,094***
$ 3,398	$ 2,875	$ 1,190	***$ 4,065***
$ 4,941	$ 2,636	$ 1,474	***$ 4,110***
$ 6,543	$ 2,460	$ 1,729	***$ 4,189***
$ 8,347	$ 2,281	$ 2,006	***$ 4,287***

Figure 11 Excel Worksheet output

7. Set up a Web server page (Figure 12) for PC Users with Excel to access the Web page from the Internet Explorer.
- Create a Web site link for installation of the VB runtime and OLE automation via a link to an MS Web site for download.
- Prepare a self-extractable .exe file from the setup package made in the above step 6, which includes a cab file, setup.exe and setup.lst. Make a link available to the user to register the program on the user's PC.
- Deploy the .vbd and cab files into a Web server and establish a link on a Web page

to the .vbd file.

Figure 12 Web server links

Summary

In this presentation, we proposed a new way of developing engineering economics course materials based on the Web. In doing so, we demonstrated an integrated cash flow analyzer based on a Java computing platform. For more complex engineering economic decision problems, we also demonstrated how to design a web interface to create an Excel file based on the user interaction. All major engineering economics problems were classified into one of two categories: Category 1—simple calculation, which can be accomplished by Java-Calculators; Category 2—requires extensive tabular presentation, suitable for Excel spreadsheet. All user interfaces were done on the web, so that students can respond to basic inputqueries. Then, the program created an Excel file ready for economic analysis. Clearly, this type of courseware development is of a worthy effort in future engineering economy instruction.

Glossary

COM	Component Object Model
DLL	Dynamic Link Library
IDE	Integrated Development Environment
IE	Internet Explorer
MARR	Minimum Attractive Rate of Return
MS	Microsoft
O&M	Operating and Maintenance
OLE	Object linking and Embedding
PC	IBM Personal Computer
VB	Visual Basic

References

1. Appleman, D. 1999. Developing COM/ActiveX Components with Visual Basic 6: A Guide to the Perplexed. Sams. Indianapolis, IN. Part IV ActiveX Documents. Pages 718 – 739.

2. Bradley, J. and A. C. Millspaugh. Advanced Programming Using Visual Basic 6. 2001. Irwin/McGraw-Hill. New York, NY.

3. Ekedahl, M. 2000. MCSD Guide to Developing Deskyop Applications with Microsoft Visual Basic 6.0 Advanced Topics. Course Technology. Cambridge, MA.

4. Green. J. 2000. Excel 2000 VBA. 1999. Wrox Press. Birmingham, B27 6BH.

5. Park, C. 2002. Contemporary Engineering Economics, 3rd Edition. Prentice Hall. Upper Saddle River, NJ. Example 15.11. Pages 729 – 736.

6. Tsay, J. 2000. Visual Basic 6 Programming: Business Applications with A Design Perspective. Prentice Hall. Upper Saddle River, NJ. Section 14.3-14.4. Automation and Creating an Active DLL. Pages 588 – 601.

7. Webb, J. 1996. Using Excel Visual Basic for Applications. 2nd Ed. Que Corporation. Hollis, NH.

CHAN S. PARK (park@eng.auburn.edu) is a professor in the Department of Industrial and Systems Engineering at the Auburn University in Auburn, AL.

JAI W. KANG (jwk@it.rit.edu) is an assistant professor in the Department of Information Technology at the Rochester Institute of Technology in Rochester, NY.