Teaching Spreadsheet-Based Decision Support Systems with Visual Basic for Applications

Susan W. Palocsay

Ina S. Markham

Office 2000 has become a standard desktop application package for employees of virtually all Fortune 500 organizations and most small and home offices. The spreadsheet component of Office 2000, Excel, is widely used throughout the business world today for processing quantitative data and developing analytical solutions. The ability to build decision support systems (DSS) based on these spreadsheet solutions can facilitate knowledge management and increase information utilization within an organization. This paper describes our experiences in preparing for and teaching a new course offering for undergraduate information systems majors in developing DSS with Visual Basic for Applications, the programming language for Excel. As a result, we hope to encourage other educators to explore this new technology as a worthwhile addition to information systems curricula.

The trend toward increasing complexity and uncertainty in the business environment, combined with the need to make decisions quickly and avoid costly errors, has changed the nature of managerial decision making at all levels of an organization (Turban & Aronson, 2001). At the same time, advances in information technology have led to more sophisticated computer software tools and greater end-user involvement in the design, implementation, and modification of applications based on these tools. The result is an opportunity to provide managers with timely information systems that can deliver the benefits promised by theoretical decision support systems (DSS).

An early definition of DSS described them as interactive computer-based systems, which help decision makers utilize data and models to solve unstructured problems (Gorry & Scott Morton, 1971). Subsequent efforts to characterize these systems have broadened the definition, but there is general agreement that the basic structure of a DSS consists of four major components: a database, model base, knowledge base, and a user interface (Turban & Aronson, 2001). The three "base" components are subsystems to store and manage corporate data, quantitative models, and organizational knowledge. The user interface provides the view of the system seen by the user and supports communication and interaction with the system.

The model component of a DSS is often derived from management science (MS) and requires the use of a mathematical algorithm for solution. While custom programming may be necessary for some highly complex models, many of these MS models can now be built using high-level software tools and modeling languages, and, most recently, spreadsheets (Savage, 1997). The enormous popularity of spreadsheets among business people has spurred the development of numerous add-in packages that embed MS solution methods in the spreadsheet environment. The first MS textbooks with spreadsheet modeling began appearing about five years ago (see Ragsdale, 1995; Camm & Evans, 1996; Hesse, 1997; Winston & Albright, 1996). Due to their influence, most MS texts have adopted some level of electronic spreadsheet usage,

Susan W. Palocsay is Professor, Computer Information Systems and Operations Management Program, College of Business, James Madison University, Harrisonburg, Virginia.

Ina S. Markham is Associate Professor, Computer Information Systems and Operations Management Program, College of Business, James Madison University, Harrisonburg, Virginia.

and the use of spreadsheets in teaching MS has become a common practice.

In retrospect, the movement away from teaching the detailed steps of algorithms toward spreadsheet-based quantitative analysis was a logical shift for MS education. Spreadsheets had become entrenched in business, driven by the demands of end users for tools to help them in performing analysis tasks. Chan and Storey (1996) found that, regardless of their level of spreadsheet proficiency, spreadsheet users are not likely to apply specialized software even if these other software packages are commercially available and potentially more appropriate for their tasks. At the same time, MS courses in business schools across the country were being reduced and, in some cases, eliminated (Willemain, 1997). Incorporation of spreadsheets into MS increased both the relevance and the popularity of the introductory MS course in business core curricula (see references in Grossman, 2001).

Most recently, Ragsdale (2001a) and Albright (2001) have independently proposed a further extension of spreadsheet-based MS models into decision support using Visual Basic for Applications (VBA), the programming language for Excel. With VBA, a user-friendly interface can be built around an MS model to provide a DSS in the familiar spreadsheet environment. The end user is not likely to be interested in the mathematical form of the model, only in its application to a specific decision task. Packaging the model with a set of dialog boxes to specify input data, and non-technical reports and charts to present the results, gives other members of an organization access to the model's functionality. Our goal was to develop a course for computer information systems (CIS) majors that would stimulate their interest in DSS by combining their knowledge of MS models with current information technology.

Background

The authors first became interested in VBA programming at the initial Teaching Management Science summer workshop at Dartmouth College in the summer of 1998, which focused on teaching MS modeling with spreadsheets. Ragsdale (2001a) and Albright (1998), two instructors at this workshop, reported positive feedback from incorporating some VBA into MS courses. They indicated, furthermore, that previous computer programming experience was not required, and students were very excited by the new computer skills they acquired in the course. The authors wondered how an introductory course in VBA with an emphasis on spreadsheet DSS might benefit their CIS majors.

A review of the IS'97 model curriculum (Davis, Gorgone, Couger, Feinstein, & Longenecker, 1997) identified three exit characteristics expected of information systems program graduates to which such a course could contribute: (1) Information Technology and Tools, which states that the graduates should be able to select and apply software tools for organizational solutions as well as develop and manage distributed systems with high-level tools and methodologies; (2) Problem Solving, which states that the graduates should be able to recognize the need for applications of analytic methods, apply systems concepts to definition and solution of problems, and formulate creative solutions to simple and complex problems; and (3) Systems Development Methodologies, which states that the graduates should be able to select and utilize appropriate methodologies; use tools and techniques to analyze, design, and construct an information systems; and apply design methodologies compatible with organizational settings. Specifically, this course could be used to help students meet objectives for selected learning units in the pre-requisite software tool kit (IS'97.P0 – Knowledge Work Software Tool Kit) and four of the ten IS'97 course descriptions (IS'97.2 – Personal Productivity with IS technology, IS'97.3 – Information Systems Theory and Practice, IS'97.5 – Programming, Data, File and Object Structures, and IS'97.7 – Analysis and Logical Design).

As companies have moved away from large mainframes in favor of smaller client-server systems, application development has moved closer to the end user. The ability to integrate these applications into components of Microsoft Office 2000 has the potential to reduce the time and cost of their development while increasing their utilization. Providing students with practical skills to use in this environment should enhance their preparation for internships and future employment.

Since learning a new technology requires both time and motivation on the part of the faculty members, the authors volunteered to develop and team-teach a new section of CIS 301, Information Technology and Tools in the fall of 2000 on VBA programming in Excel, as an overload. CIS 301 is a 1-credit, hands-on course required for all CIS majors, with an objective of giving CIS students experience with management productivity tools on microcomputers. The specific software taught under this course number varies.

This arrangement was attractive to the authors, as well as their program director, for several reasons. It would provide an additional section of a course that was in high demand while permitting the authors to cover their regular teaching responsibilities (9 credit hours per semester) at the same time. The only additional resource needed from an administrative point of view was classroom space, and it was customary to teach CIS 301 in the computer lab. Using the computer lab had the added advantage of restricting the class size to a maximum of 25 students. The team teaching aspect was appealing since this would be the authors' first experience with VBA programming and there were no other faculty in their program with VBA/Excel skills. Their liability would also be limited to preparing for one hour of class per week, and they could assume that every student had previously taken the first- and second-year business core classes, including an introductory management science course required by all business majors.

Course Description

The first step in preparing to teach the course was to identify resources for use by the authors in learning VBA programming and a text for use by the students in the class. Copies of the references listed in Table 1 were obtained, with the exception of Albright's book (since it was not yet published). The authors selected *Microsoft Excel 2000 Visual Basic for Applications: Fundamentals* by Jacobson (1999) as the text for the course. This choice was based on the tutorial nature of the Jacobson book as well as its coverage, which was limited to introductory Excel VBA programming. Jacobson has developed a series of short lessons that cover important topics in VBA, each consisting of a number of practical exercises that can be used either for demonstration purposes or for interactive learning. The instructions for each exercise are clearly written and accompanied by annotated screen illustrations, providing a straightforward introduction to even the most difficult concepts in VBA programming.

To add a MS modeling component to the VBA course, the authors turned to the recent work of Albright (2001). This material discussed the VBA functions available to manipulate the Solver optimization add-in (included in Excel) and gave an example of a DSS for a typical oil-blending linear programming (LP) model. Frontline Systems, Inc., the company that provides Solver to Microsoft, also has good documentation of these VBA functions on their website at http://www.frontsys.com/mlvbaref.htm. Students were provided with this link to obtain the reference material they needed for controlling Solver from VBA.

Class sessions were scheduled in the computer lab for one 50-minute period per week, with 14 class meetings over the semester. Although the students enrolled were CIS majors, the authors were concerned about disparities in knowledge of Excel spreadsheets and in degree of programming experience among the students. A short questionnaire was administered during the first class meeting asking students to rate their spreadsheet skills, to indicate if they had used Solver in their MS

Table 1: List of VBA Books

Albright, S. C. (2001). *VBA for modelers: Developing decision support systems with Microsoft Excel.* Pacific Grove, CA: Duxbury Press.

Boctor, D. (1999). *Office 2000 Visual Basic for applications: Fundamentals*. Redmond, WA: Microsoft Press.

Green, J., Bullen, S., and Martins, F. (1999). *Excel 2000 VBA programmer's reference*. Birmingham, UK: Wrox Press.

Jacobson, R. (1999). *Microsoft Excel 2000 Visual Basic for applications: Fundamentals*. Redmond, WA: Microsoft Press.

Lomax, P. (1998). *VB & VBA in a nutshell*. Sebastopol, CA: O'Reilly & Associates, Inc.

Walkenbach, J. (1999). *Microsoft Excel 2000 power programming with VBA*. Foster City, CA: IDG Books Worldwide, Inc.

Zak, D. (2001). *Visual Basic for applications*. Cambridge, MA: Course Technology.

course, and to list all of the programming courses they had previously taken or were taking in the current semester. The survey results are summarized in Table 2.

To create a common foundation in basic Excel skills, an interactive tutorial written by Albright (available from http://www.indiana.edu/~busk410/ExcelTutorial.html) was assigned to students at the first meeting. Students were also given a set of written questions about the most important topics in the tutorial.

An overview of the course design is provided in Table 3. The authors found this to be a very effective way to deliver the course material as it significantly reduced the amount of class lecture time and allowed the students to discover important characteristics of VBA on their own. Since the solutions to the text exercises were included on a CD-ROM with the book, "checkpoints" were collected to require students to complete additional tasks that augmented or extended the book's exercises. At the end of each class, students were given a checkpoint or take-home assignment reinforcing the new material to be completed before the next class meeting. Table 4 lists the topics covered in this course.

Starting at week 9, the authors shifted the focus of the class from "programming" to "application development" by relating VBA programming concepts to MS models. A useful example of a user-defined function with arguments from MS is one that computes P_0, the probability of no customers in a multi-server queuing system (Anderson, Sweeney, & Williams, 1999). It is not possible to write a single Excel formula that will handle any number of servers since a summation term in the formula is indexed by this number. This example also provided an opportunity to emphasize the importance of error checking when developing modules for others to use. In this case, the problem occurs when the mean arrival rate is less than the product of the number of servers and the mean service rate (i.e., the average system utilization is less than one). A VBA procedure was created to check for this condition before calling the function, and to notify the user (Figure 1).

Table 2: Summary of Background Data

- Excel expertise on a scale of 1 (beginner) to 10 (very advanced)
  - 7.65% indicated level 5
  - 7.65% indicated level 6
  - 46.2% indicated level 7
  - 38.5% indicated level 8
- Previous use of Solver
  - 61.5% indicated Yes
  - 38.5% indicated No
- Previous knowledge of Visual Basic
  - 30.77% indicated Yes
  - 69.23% indicated No
- Previous knowledge of Visual Basic for Applications
  - 23% indicated Yes
  - 77% indicated No
- Previous knowledge of other programming languages
  - 22.7% indicated None
  - 77.3% indicated Some, specifically
    - 7.65% Fortran and Pascal
    - 7.65% C++ and Java
    - 31% COBOL
    - 31% COBOL and C

Table 3: Overview of Course Design

Format of each class
- Overview of material
- Demonstrations as appropriate
- Hands-on exercises

Grade allocation
- Checkpoints accounting for 20% of grade
- Take-home assignments in lieu of tests
  - Assignments I and II each worth 20% of grade
  - Assignment III worth 40% of grade

Table 4: Overview of Course Content

Week	Topic
1	Excel Tutorial
2	Recording and running Macros; Using Visual Basic Editor
3	Design issues of macros; Relative referencing
4	Excel Object Model; Object Browser; Auto Lists
5 & 6	Range Objects
7	Chart Objects
8	Loops and conditional statements
9	Custom functions
10 & 11	Formulating and solving linear programming (LP) models
12 - 14	ActiveX controls; User forms; Creating a Decision Support System (DSS)

Figure 1: VBA function for queuing

```
Function P0(k, lambda, mu)

   Sum = 0

   For n = 0 To k - 1
      Sum = Sum + (lambda / mu) ^ n / WorksheetFunction.Fact(n)
   Next n

   P0 = 1 / (Sum + ((lambda / mu) ^ k / WorksheetFunction.Fact(k)) * (k * mu / (k * mu -
      lambda)))

End Function
```

```
Sub MultiServerP0()

   Dim NumServers As Long
   Dim ArrivalRate As Long
   Dim ServiceRate As Long

   NumServers = InputBox("Enter the number of servers: ")
   ArrivalRate = InputBox("Enter the mean arrival rate: ")
   ServiceRate = InputBox("Enter the mean service rate for each server: ")

   Range("B7").Value = NumServers
   Range("B8").Value = ArrivalRate
   Range("B9").Value = ServiceRate

   If ArrivalRate < NumServers * ServiceRate Then
      Range("C14").Value = P0(NumServers, ArrivalRate, ServiceRate)
   Else
      MsgBox ("The mean arrival rate must be slower than the mean service rate of the
            system")
      Range("C14").Value = "#NA"
   End If
End Sub
```

The final assignment required students to develop an interactive LP model-based DSS. The problem scenario was a manufacturing company with a typical make-buy decision problem for two products, electric and gas lawn trimmers (Ragsdale, 2001b). The data and formulas for the LP model were maintained on one worksheet that is hidden from the user, as shown in Figure 2. Additional worksheets were created to give an overview of the application (Figure 3), report the results of solving the model (Figure 4), and show charts of the optimal production and purchasing mix (Figure 5) and resource usage (Figure 6). Command buttons, a type of ActiveX control, were added to appropriate worksheets to run the application and to view and navigate between the sheets. Custom dialog boxes, referred to as user forms in VBA, were designed and implemented with event code to allow the user the option of modifying data in either the objective function or constraints of the LP model. The main VBA code for the application is responsible for showing the forms, setting up the Solver dialog box, and running Solver to obtain the optimal solution. Various error-checking procedures were also included to verify user inputs, identify special cases such as infeasibility, and ensure that the system operated properly. A copy of the Excel file for this application can be obtained from the authors.

Pedagogical Issues

Because the course was taught in a computer lab, the authors were able to de-emphasize the lecture presentation of technical material and directly involve the students in hands-on experiential learning activities. This approach had many positive benefits, and the authors are not sure how the transfer of the course to a traditional classroom setting or an increase in class size would affect the overall results. The format allowed the authors to see firsthand where students were encountering learning difficulties with the new material and offer immediate assistance based on the needs of either individual students or the class as a group.

Figure 2: Linear programming model sheet

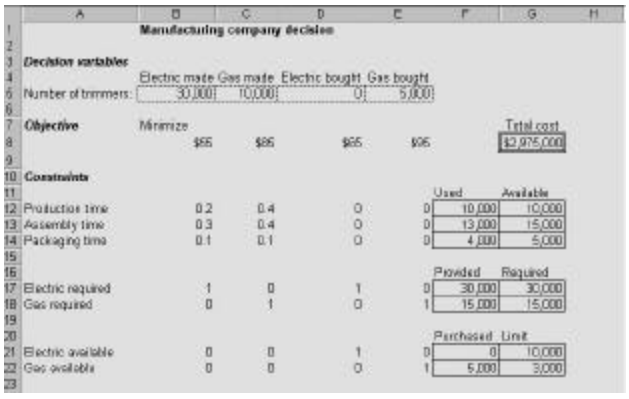


Figure 3: Explanation sheet for application

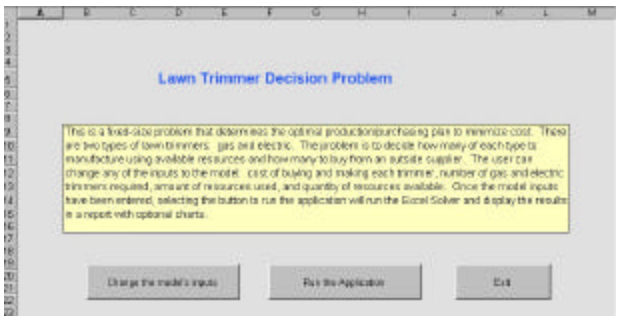


Figure 4: Report of model results

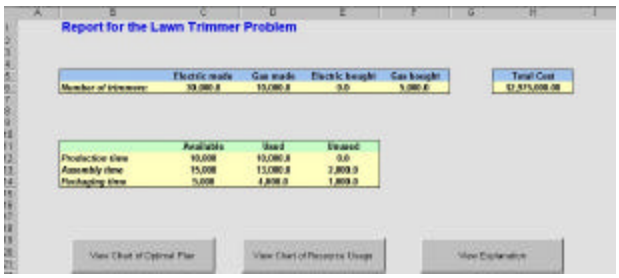
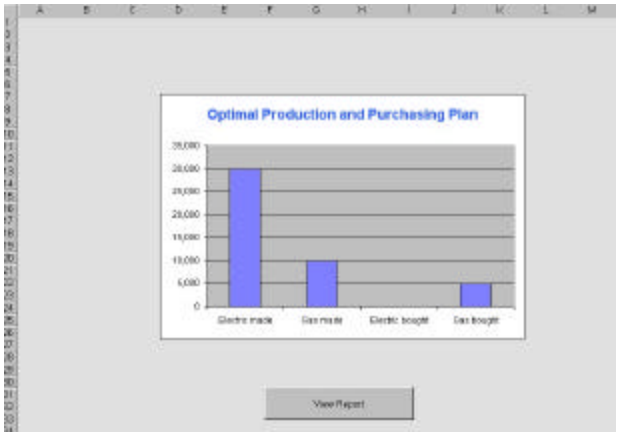


Figure 5: Chart of optimal production/purchase mix



For example, it was easy to illustrate and discuss the differences in code generated by the Recorder (FormatPercent, in Figure 7) and one created by the user (FormatPercent2, also in Figure 7). Unfortunately, the Macro Recorder often generates a significant amount of unnecessary code, and there are some actions and logic that cannot be recorded. Students were quick to see the elegance and usefulness of a shortened version with the same functionality.

Although there was considerable computer programming experience among the students in the class, there were weaknesses in several areas. One area is the concept of object orientation, where an object such as a range of cells on a worksheet is selected and then changes are made in its properties or methods are applied. The primary difficulty seemed to be more in identifying and selecting the proper object than in manipulating the object. Range objects seemed to be the most difficult concept in VBA programming for students to grasp, regardless of their level of programming experience. By definition, a range consists of a set of one or more cells on a worksheet, and much of the functionality of a VBA procedure involves selecting a range of cells, changing its characteristics (i.e., properties), and performing actions (i.e., methods) on it. For the purpose of exploring Range objects, the class was asked to predict the cells that would be affected by each VBA statement in a procedure before its execution (see Range 1 and Range 2 in Figure 8). Changing the color of selected cells helped students to visualize the results of manipulating a range. To effectively use Range objects in VBA programming, the concept of storing a *reference* to an object, in contrast to storing a *value* in a variable, must be well understood.

Students also struggled with traditional programming concepts, especially conditional logic (e.g., If...Then...Else statements) and sequential processing (e.g., For loops). Students found the demonstration of a For loop with a macro to compute a 3-period moving average to be helpful.

Figure 6: Chart of resource usage

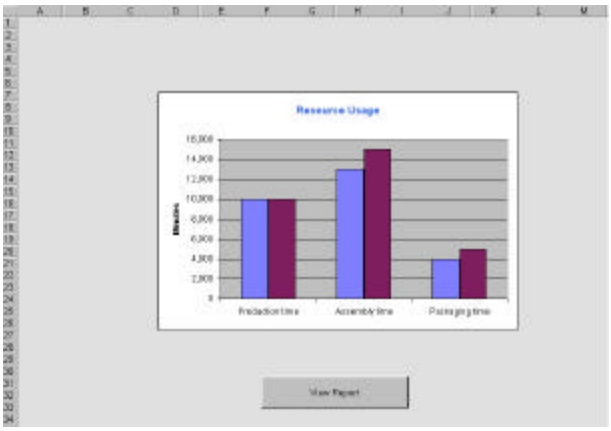


Figure 7: Example of recording and editing a VBA macro

```
Sub FormatPercent()
'
' FormatPercent Macro
' Macro recorded 7/8/2001
'
' Keyboard Shortcut: Ctrl+Shift+P
'
    ActiveCell.Select
    Selection.NumberFormat = "0.0%"
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .ShrinkToFit = False
        .MergeCells = False
    End With
End Sub
```

```
Sub FormatPercent2()

' Edited version of FormatPercent macro

    With Selection
        .NumberFormat = "0.0%"
        .HorizontalAlignment = xlCenter
    End With

End Sub
```

Figure 8: Manipulating Range objects

```
Sub Range1()
    Dim myRange As Range
    Set myRange = ActiveCell.CurrentRegion
    myRange.Interior.Color = vbCyan
    myRange.Rows(4).Select
    myRange.Rows(myRange.Rows.Count).Select
    myRange.Columns(2).Select
    myRange.Columns(myRange.Columns.Count).Select
    myRange.Cells(2, 3).Select
    myRange.Cells(11).Select
    myRange.Cells(myRange.Cells.Count).Select
    myRange.Cells.ClearFormats
End Sub

Sub Range2()
    Dim myRange As Range
    Set myRange = Range("A5:B6")
    myRange.Interior.Color = vbYellow
    myRange.Offset(0, 1).Select
    myRange.Resize(, 3).Select
    myRange.Resize(3).Select
    myRange.Offset(-4, 0).Resize(myRange.Rows.Count + 6,
myRange.Columns.Count + 1).Select
    myRange.EntireColumn.Select
    myRange.Cells(3).EntireRow.Select
    myRange.Cells.ClearFormats
End Sub
```

As an exercise, the students extended this macro to ask the user for the range of input data and the number of periods for the forecasting analysis (Figure 9). There was also a discussion of the advantage of writing a formula into a cell that will automatically be updated if there are changes in the input data, rather than writing only the calculated value as Excel's built-in Moving Average tool does. In addition to the For, Do Until, and Do While looping structures, VBA also offers a For Each loop that provides a convenient way to process a group of objects belonging to a collection, such as all of the worksheets in a workbook. An example of the use of this construct in a DSS, suggested by Albright (2001), is to include a procedure like the one shown in Figure 10 to hide all of the worksheets except an introductory explanation sheet when the workbook is opened.

Another serious deficiency that was prevalent among students in the class was the lack of debugging skills. Students were encouraged to take advantage of the tools provided by VBA for debugging such as the ability to execute the code one line at a time, by repeatedly pressing a function key, and immediately viewing the effects in the Excel workbook. Students were often observed to be too impatient to step through their procedures and analyze the performance of their code. Another concern was that students did not apply techniques such as commenting out a line or inserting write (i.e., MSGBOX) statements to help them debug their code. In revising the course, the authors recommend looking for ways to incorporate more discussion and practice on debugging skills.

A final pedagogical issue was the balance between standardization and creativity on the part of the students in programming. As a result of the numerous properties and methods associated with VBA objects in Excel, there are many ways to write VBA code that will perform a certain set of operations. The authors discovered early on in the course that instructions for assignments had to specify which properties and methods to use, if the objective was to give the students practice with particular ones. For some assignments, however, these instructions were deliberately omitted so that students would be forced to think about the various objects and use their book, notes, and/or online help to find appropriate members of the objects to accomplish the required tasks. As a result, the grading of these assignments was considerably more

Figure 9: Moving average forecasting

```
Sub MovingAverage()

    Dim myRange As Range
    Dim myMove As Range
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer
    Dim myPeriod As Integer

    Set myRange = Range(InputBox("Enter the range of sales for the moving average in the form B1:B#"))

    myRange.Offset(0, 1).EntireColumn.ClearContents

    myPeriod = InputBox("Enter the number of periods for the moving average")

    Set myMove = myRange.Resize(myPeriod)

    j = myPeriod + 1
    k = myRange.Rows.Count + 1

    For i = j To k
        Cells(i, 3).Value = WorksheetFunction.Sum(myMove) / myPeriod
        Cells(i, 3).NumberFormat = "#,##0.00"
        Set myMove = myMove.Offset(1, 0)
    Next i

End Sub
```

difficult and time consuming, and required actually running students' procedures rather than reviewing printed code to verify that they work properly. Students were initially uncomfortable with these less-structured assignments, but gradually came to value the confidence they gained from them as the course progressed.

Future Plans

A formal course evaluation (summarized in Table 5) was administered on the last day of class. Every student commented that the workload was very high relative to the number of credits earned (i.e., one). Ninety-five percent of the students also indicated that they learned material perceived by them to be of use in the future. Fifty percent of the students recommended that the class should meet a couple of times each week. The authors agreed with this assessment, and the course will be offered as a 3-credit special topics elective course that meets twice a week in the future.

This format will allow time to expand coverage of VBA topics in the initial course outline, and to explore more capabilities of VBA, such as building DSS based on other types of MS models and linking Excel spreadsheets to other Office 2000 components. Albright (2001) includes some excellent examples of DSS for simulation and forecasting applications. Finance and accounting are additional sources of potential applications for spreadsheet-based DSS. Since VBA is also the programming language for Word, PowerPoint, and Access, one possible extension of the course is in the direction of developing integrated DSS in Office to link spreadsheets with documents, presentations, and databases. An example DSS for portfolio optimization where the data are stored in an Access database and queried from Excel using SQL commands, described in Ragsdale (2001a), illustrates this approach for advanced DSS.

Interestingly, one of the students worked on an independent project that built on the material from the course. This student has since graduated and is working with a well-known consulting firm. This

Figure 10: For Each loops

```
Private Sub Workbook_Open()

    Dim mySheet As Object

    Worksheets("Explanation").Activate
    Range("A1").Select

    For Each mySheet In ActiveWorkbook.Sheets
        If mySheet.Name <> "Explanation" Then mySheet.Visible = False
        Application.ScreenUpdating = False
    Next

End Sub
```

Table 5: Course Evaluation Questionnaire and Student Comments

1. What did you like best about the course?
   - *Material was neat/useful/helpful/relevant*
2. What did you like least about the course?
   - *Too much work for 1 credit*
3. What do you think we could do to improve the course?
   - *Make it a 2 or 3-credit course*
   - *Meet twice a week*
4. Overall, what was your impression of the course?
   - *Too much work*
   - *Great learning experience*
   - *Learned valuable skills*

fact, along with the comments from student evaluations, suggests that the VBA course with emphasis on DSS does benefit CIS majors. The authors believe that VBA gives IT educators an excellent opportunity to teach students in information systems how to create applications that interface with data and models and perform quantitative analysis. The widespread use of Office 2000, and Excel in particular, promises that these skills will be immediately applicable in the business environment and directly contribute to the efforts of companies that are focusing their efforts on better utilization of these tools. The authors' initial experience teaching VBA programming to CIS majors lead them to recommend that IT educators add VBA to their curricula as a way to take advantage of modern spreadsheet technology and apply this technology in the context of business organizational needs.

References

Albright, S. C. (1998). Using VBA in a management science course. *OR/MS Today*, *25*(3), 6.

Albright, S. C. (2001). *VBA for modelers: Developing decision support systems with Microsoft Excel*. Pacific Grove, CA: Duxbury Press.

Anderson, D. R., Sweeney, D. J., & Williams, T. A. (1999). *Contemporary management science with spreadsheets*. Cincinnati, OH: South-Western College Publishing.

Camm, J. D., & Evans, J. R. (1996). *Management science modeling, analysis and interpretation.* Cincinnati, OH: South-Western College Publishing.

Chan, Y. E., & Storey, V. C. (1996). The use of spreadsheets in organizations: Determinants and consequences. *Information & Management*, *31*(3), 119-134.

Davis, G. B., Gorgone, J. T., Couger, J. D., Feinstein, D. L., & Longenecker, Jr., H. E. (1997). *IS'97 Model curriculum and guidelines for undergraduate degree programs in information systems*. Park Ridge, IL: Association of Information Technology Professionals.

Gorry, G. A., & Scott Morton, M. S. (1971). A framework for management information systems. *Sloan Management Review*, *13*(1), 55-70.

Grossman, Jr., T. A. (2001). Causes of the decline of the business school management science course. *INFORMS Transactions on Education, 1*(2), 51-61.

Hesse, R. (1997). *Managerial spreadsheet modeling and analysis*. Chicago: Richard D. Irwin, a Times Mirror Education Group, Inc. Company.

Jacobson, R. (1999). *Microsoft Excel 2000 Visual Basic for applications: Fundamentals*. Redmond, WA: Microsoft Press.

Ragsdale, C. T. (1995). *Spreadsheet modeling and decision analysis, 1st Edition.* Boston: Course Technology, Inc.

Ragsdale, C. T. (2001a). Teaching management science with spreadsheets: from decision models to decision support. *INFORMS Transactions on Education, 1*(2), 68-74.

Ragsdale, C. T. (2001b). *Spreadsheet modeling and decision analysis, 3rd edition*. Cincinnati, OH: South-Western College Publishing.

Savage, S. (1997). Weighing the pros and cons of decision technology in spreadsheets. *OR/MS Today*, *24*(1), 42-45.

Turban, E., & Aronson, J. E. (2001). *Decision support systems and intelligent systems.* Upper Saddle River, NJ: Prentice-Hall.

Willemain, T. (1997). OR/MS and MBAs: Mediating the mismatches. *OR/MS Today*, *24*(1), 36-41.

Winston, W. L., & Albright, S. C. (1996). *Practical management science, 1st Edition.* Pacific Grove, CA: Duxbury Press.