Vol. 5, No. 2, January 2005, pp. 9–16
ISSN 1532-0545 | 05 | 0502 | 0009
INFORMS Transactions on Education
DOI 10.1287/ited.5.2.9

Using Multimedia to Facilitate Software Instruction in an Introductory Modeling Course

Jill R. Hardin
Department of Statistical Sciences and Operations Research, Virginia Commonwealth University, Richmond, Virginia 23284, jrhardin@vcu.edu

Aimee J. Ellington
Department of Mathematics and Applied Mathematics, Virginia Commonwealth University, Richmond, Virginia 23284, ajellington@vcu.edu

Multimedia tutorials were developed to introduce basic software functions to students in an introductory operations research modeling course. The purpose of these tutorials was to reduce the amount of class time spent addressing deficiencies in software mastery and allow for more class time to focus on modeling concepts. The motivation for developing these tools and details on the design of the tutorials are outlined. We conducted a quasi-experimental pre-test/post-test assessment of the value of the tutorials. The results of the formal assessment as well as student reactions and anecdotal evidence of their usefulness are discussed. We conclude with an evaluation of lessons learned in the process and encourage other educators to consider a similar approach.

1. Introduction

Mathematical Modeling is a junior-level course requirement at the Virginia Commonwealth University for students majoring in mathematics or operations research, as well as a state certification requirement for mathematics education majors. It is also chosen as a technical elective by many engineering students. In the fall of 2002, Mathematical Modeling was modified to give students a one-semester introduction to the wide assortment of modeling techniques that are used by operations research analysts. The course now uses spreadsheet modeling to give students practice with and understanding of the wide variety of operations research models at work in the world around them.

Mathematical Modeling in its new format was developed jointly with Dr. Jason Merrick of the Department of Statistical Sciences and Operations Research, and was taught for the first time in the fall of 2002. The course aims to focus on constructing, validating, and analyzing models, without focusing excessively on the specifics of solution methods. Nevertheless, we want to give our students experience with interpreting and applying model solutions. We note that even experienced modelers often find it necessary to obtain solutions as a means of model validation. As such, our Mathematical Modeling course incorporates the Microsoft Excel add-ins that accompany the course textbook, Practical Management Science (Winston and Albright 2001), as tools for obtaining and analyzing solutions to spreadsheet models. With such an academically diverse group of students enrolled in the course, however, teaching how to use these software add-ins proved to be quite a challenge. During the fall of 2002 and subsequent spring semester, we observed that the class time required to help students become proficient in the use of the software add-ins precluded adequate discussion of the modeling issues and applications themselves. The result was that students left the course with some mastery of solving models using appropriate software, yet they had insufficient expertise in developing them. We hoped that if students had additional help with the course software outside of class, then more instructional time could be spent on model construction and validation.

These observations raised two important questions. First, would requiring students to view a tutorial prior to using a software package help to alleviate our problems with software instruction? If so, should we use tutorials already available, or create our own? After much deliberation, we chose to develop our own tutorials so that we could specifically target the needs of our students and the objectives of our course. Through the support of an equipment grant from

the VCU Instructional Development Center, we developed a set of online multimedia tutorials that would demonstrate the course software to students, introducing them to basic functions and allowing them to solidify their familiarity with the software outside of class. In addition to adding variety to the supplemental course materials available, use of these tutorials would allow us more class time to focus on conceptual understanding of model development.

In what follows, we discuss our rationale for the choice to create our own modules, describe their development, and assess the effectiveness of the resulting product. We begin our discussion by describing the existing Mathematical Modeling course format and situating our paper in the context of existing research. Justification of our choice to create our own tutorials is followed by a description of their design and implementation, including a discussion of available software for creating modules of this kind. We then provide a formal assessment of their value as well as anecdotal evidence of their utility. We conclude with a discussion of lessons learned in this process and a word of encouragement to other educators facing similar pedagogical issues, even in courses outside management science.

2. Motivation

In the summer of 2002, responsibility for the Mathematical Modeling course was placed in the hands of operations research faculty. In order to introduce more students to the possibilities allowed by operations research approaches, the faculty decided to modify the course to focus on spreadsheet models in four basic operations research areas: linear and integer programming, network modeling, decision analysis, and simulation. The course is now taught in the department's computer classroom, which is fitted with high-end computers for students and faculty, computer projection, and video and cable input. Additionally, all students enrolled in the course are given access to a computer lab whose machines have software and desktop setup identical to the machines in the classroom. Students have access to this lab any time the classroom building is open, including evening hours.

As this is an introductory course targeting a diverse group of students, the course focuses on understanding, building, and analyzing models rather than on the specifics of various solution techniques. We reinforce this approach by encouraging hands-on model development, rather than by lecturing extensively. Our access to computer classrooms makes this strategy possible. While students are presented with an intuitive explanation of how model solutions are found by the software packages they use, we do not expect them to learn the details of the underlying algorithms.

Class time is typically spent working through models—as a class, individually, or in small groups. Some lecture is occasionally necessary, especially when introducing new classes of models. Students must be oriented to the new type of model through description of basic characteristics and development of introductory instances. Once this introduction has been made, however, class time is spent working through models and answering questions about them. A schedule is posted on the course website indicating which textbook problems will be addressed in each day's class. While it rarely happens, students have the opportunity to attempt to build models on their own before coming to class. Initially, students work through the development of a model along with the instructor. Once students have some familiarity with the new type of model, they are given the freedom to attempt a new problem on their own or in conjunction with one or two classmates. Students rapidly see that constructing models from scratch on their own is much more challenging than simply "following along." Making these attempts in class, however, allows the instructor to identify mistakes and deficiencies before homework problems are assigned. We further encourage students to be involved in these classroom activities by making class participation a substantial part (15%) of the course grade.

To reinforce what they learn in class, students are required to submit one to two spreadsheet models per week as homework. This forces students to build models on their own, even if they have not been active participants in the classroom. Solutions to all group exercises and homework problems are posted on the course website, allowing students to review any exercise as needed. Examinations combine written answers to questions with hands-on spreadsheet model development. We should note that, because students have ample access to a functional computer lab and all course-required software, they are hard-pressed to blame computer problems for their inability to complete homework or to review posted solutions.

Since we spend so much class time formulating models and such a large portion of student assessment is based on aptitude in this task, it is important that students become competent in the basic use of solution software very quickly. As discussed previously, however, the broad range of academic programs represented by students who typically enroll in the class results in a group of students with widely varying computer competencies. Many are very comfortable with learning new computer-oriented skills, while others struggle with these tasks. In previous semesters, when students struggled with basic computer functions, much class time was consumed by reviewing software tasks; moreover, it seemed

that the same straightforward questions about basic software functions were answered repeatedly. "How do I create a new tree with PrecisionTree?" "How do I start my @RISK simulation?" These repetitive questions—while legitimate sources of confusion—were consuming valuable class time and reducing the amount of time available for hands-on modeling experience. Addressing deficiencies in software mastery prevented adequate discussion of the modeling issues and applications the course was designed to address. As a result, at the end of the course, students were proficient in using software to solve models but did not have enough experience in model development and analysis. We hoped that more instructional time could be spent on these areas if students had an out of class resource for basic software-related questions. Troxell and Aieta (Troxell and Aieta 2000 and Troxell 2002) developed PowerPoint tutorials for the Solver add-in to address similar pedagogical concerns. We took this idea a step further and introduced online multimedia tutorials, utilizing video screen capture and audio voice-over, to address this issue.

With the plethora of software tutorials available online, why develop our own tutorials rather than utilize existing ones? None of the tutorials accessible to our students covered the information we needed at the level we thought appropriate for an introductory course. Our students have access to SmartForce http://www.smartforce.com/learning_community/, a web-based software training system from SkillSoft that offers over 300 tutorials, or "smartcourses." While several tutorials on Microsoft Excel are included, none of them offer instruction on the use of the add-ins that we utilize in the course. We do, however, refer students to Smartforce to remedy any basic Excel deficiencies. Albright et al. (2001) also list an Excel tutorial on the textbook website, but again this tutorial focuses on Excel functions rather than software add-ins.

Palisade Corporation, which distributes both PrecisionTree and @RISK—two of the add-ins that we use—provides tutorials for both of these software packages on its website http://www.palisade.com/html/index.html. These tutorials seem to be designed to sell the merits of the software to potential users, rather than to instruct in the basic features of the software for undergraduate students. The PrecisionTree tutorial, for example, builds a multi-stage decision model in order to demonstrate basic tree-building functions. For students who have never seen a decision tree such a complex model can be overwhelming. These tutorials also cover advanced topics and uses of the software packages. We preferred that our students be given a very basic introduction to the purposes and functionalities of the software, showing them just enough to get started. We can cover more advanced features in class, once they have become familiar with the models and their basic implementation.

We are aware that others have developed tutorials that focus more on the instruction of basic software features than on advanced model development. Terence Reilly, for example, has utilized such tutorials in his Babson College courses for several years now. These tutorials are largely text-based and teach software via model development, rather than focusing on first steps. While very well constructed, they did not achieve the purpose we envisioned for our tutorials. Their existence, however, shows that others are testing a similar approach. It is our understanding that Reilly is modifying his tutorials, and an updated version will eventually be included with several Duxbury texts. The fact that textbook publishers are beginning to include such tutorials aimed at beginning modelers makes our assessment of their effectiveness all the more timely. Furthermore, with the rapid development of software improvements, textbook publishers will be hard pressed to provide supplementary instruction for new features. Having the capability to address software features almost instantly makes the development of our own tutorials worthwhile.

3. Related Literature

Many OR/MS educators have lobbied for change in the way that operations research is taught, encouraging less reliance on lectures and more reliance on active and independent learning (Liebman 1994 and Belton and Scott 1998, for example). They have also advocated the need to work with the skill sets that students bring to the classroom, no matter how lacking. Grossman (2001) goes so far as to say that "poor student mathematical preparation is not an acceptable justification for poor teaching outcomes." Extending this notion to software instruction, student deficiencies in computer skills should not be an excuse for failure to teach students modeling skills. We must find ways to overcome these deficiencies and multimedia opportunities made available by the PC revolution may provide some help. They can not only be used to teach new skills, but they can also cater to a wider variety of learning styles (Wiest 2001).

The use of computer-assisted learning is not new in quantitative subjects. There are many examples of online modules or tutorials designed to enhance instruction, however most of these were developed to teach concepts rather than software tools (see Belton et al. 1997, Wood 1996, Callahan et al. 2000, and Wahl 1995). Furthermore, literature describing such endeavors—including instructional supplements that support software directly (Daku and Jeffrey 2001, Troxell and Aieta 2000, and Troxell 2002)—provide

only qualitative assessments of their effectiveness, an observation echoed by Crowe and Zand (2001). Hence, there is a need for quantitative analysis of student experiences with online tutorials.

Other OR/MS educators have expressed concern over the potential of software instruction to preclude development of modeling proficiency. Powell (1995, 2001) writes that developing skill with software tools should take a backseat to problem-solving and modeling capabilities, and Gass et al. (2000) state their belief that "striving to get the spreadsheet right is taking precedence over learning what is right in modeling." Others (Seal and Przasnyski 2003 and Troxell and Aieta 2000) have also noted that, in their own courses, too much class time must be spent on teaching tools or software, detracting from concentration on OR/MS concepts. We developed online instructional modules to address this concern.

4. Module Design

Three software add-ins were addressed in online tutorials: Solver, PrecisionTree, and @RISK. We created an additional module to instruct students in building Excel data tables, in response to deficiencies with this task. We note that the Excel tutorial by Albright et al. (2001) addresses the use of data tables, however, for consistency, we chose to create our own instructional module for this task.

Modules were created using the Microsoft Producer add-in to PowerPoint. This freely available download allows the display of PowerPoint slides accompanied by video screen capture and audio voice-over. PowerPoint slides are used to outline and emphasize key points and concepts, while video screen capture allows a visual demonstration of the software. The audio component ties these elements together with more detailed explanations and discussions. Many other software products—Breeze http://www.macromedia.com/software/breeze/ (by Macromedia), Screencorder http://www.matchware.net/en/products/screencorder/default.htm (by MatchWare), Camtasia http://www.techsmith.com/products/studio/default.asp?lid=CamtasiaStudioHome (by TechSmith), and PresentationMaker http://www.realnetworks.com/products/presentation/ (by RealNetworks), to name a few—are available for creating multimedia tutorials such as these. Most of these packages yield results that are far more polished than those we were able to create with Microsoft Producer, yet this improvement comes at a price ranging from around $150 to nearly $2,000. As our instructional grant included only equipment and standard software and with no budget for purchasing a multimedia authoring package, Microsoft Producer was an obvious choice.

Modules were designed to focus on basic software usage rather than on modeling concepts or on advanced software skills. We intended, however, for students to watch the tutorials before related models were discussed in class so that they would already be familiar with the accompanying software. As such, it was necessary to present a very simple model and explain its characteristics and development briefly in order to address how the related software add-in could be used to build the model. Tutorials spend very little time explaining modeling rationale, but instead give a short explanation of each model component and move on to software solutions.

Design of each module began with the identification of the particular functions students needed to learn for each add-in in order to construct basic models. Next, we "scripted" each module, a suggestion we gleaned from Seal and Przasnyski (2003), determining the flow of the material, the manner in which each function should be demonstrated, and what narration should accompany each demonstration. Once scripting was complete and accompanying PowerPoint slides were designed, we recorded the voice-over, coordinating slide timing. Video screen capture was the last step in the process. All four modules follow a similar format, and complete viewing of any module requires less than ten minutes.

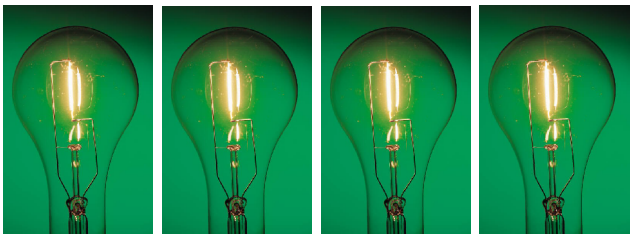
Since the development of the four modules was similar, we present the contents of the multimedia tutorial for the Solver add-in in detail, noting information about the other modules as necessary. It is assumed that, when students first view this module, they have had no instruction in the use of the Solver add-in, or even on the definition of a linear program. As such, the module begins with a basic description of a linear program, simply noting that an LP aims to optimize some objective subject to one or more restrictions. While the example we present is indeed linear, we save discussion of linear models versus nonlinear models for class in order to keep the module as short as possible and to maintain focus on software as much as possible. Once the basic aim of an LP is described, a modeling scenario is presented. This scenario presents a two-variable, two-constraint diet problem. Once the details are given, the tutorial illustrates to students an appropriate spreadsheet set-up. First, they are instructed to select changing cells, followed by a description of the quantities that changing cells will represent. The module then walks them through the process of setting up formulas to represent nutritional content of the resulting menu, which will form the left-hand-sides of their eventual constraints. Finally, students are instructed to select a target cell, which will represent the quantity they are trying to optimize. In this case, they are trying to minimize the cost of a diet that meets all nutritional

requirements. In order to optimize this quantity, they must express the cost in terms of the changing cells. The tutorial explains in detail how to establish this formula.

Once an appropriate spreadsheet model has been developed, the tutorial then demonstrates how to open Solver. In this case, since a version of Solver is part of standard Excel installation, students simply need to be shown how to access it. The tutorials for PrecisionTree and @RISK assume that students have already used the CD accompanying the course textbook to install the software add-ins. The tutorial simply shows them how to access and open the given add-in package.

When Solver is opened, the Solver Dialog Box appears. Students are taken through the steps of selecting optimization sense (max or min), selecting the target and changing cells, and entering the appropriate constraints. Specification of integer and binary variables are not covered at this time, since we want students to become familiar with the basic features of Solver necessary to solve an LP. We cover specification of integer and binary variables in Solver later in the course as students learn to develop integer programming models. Next, students are shown how to open the Solver Options Dialog Box and how to specify nonnegativity and linearity. As these are options they will need to use often, we felt it was important to include these from the very beginning.

Finally, students are shown how to solve the problem. The tutorial also discusses where students can find the optimal solution. While this is rather straightforward in Solver, more discussion is required for the other add-ins. With the PrecisionTree module, it is especially important to explain that no "solution button" exists. The solution is displayed the instant the model is complete. Similarly, they must be told that @RISK will not display a "right" answer. It is up to them to interpret and compare the simulation results for two or more decision alternatives in order to make a decision. Click below to view the resulting multimedia modules.



Modules were made available online for easy access. Once posted, they were available for the remainder of the semester. Students were allowed to view the modules as many times as needed to firmly grasp the concepts addressed. It is worth noting that students required a high-speed internet connection in order to view the modules from home. The slow speed of dial-up modem connections made viewing all but impossible. Access to an on-campus computer lab with all course-required add-ins, however, still allowed students to utilize these modules to their fullest extent.

5. Assessment

While one informal measure of the multimedia tutorials' effectiveness is the decrease in class time spent on software issues as noted by the course instructor, we desired to have a more formal measurement of the usefulness of these new instructional tools. We pursued this evaluation through the use of pre- and post-testing.

We found that use of the multimedia modules did indeed decrease instructional time spent on software tasks, yet we can provide only anecdotal evidence. When Mathematical Modeling in its new format was offered for the first time, in the Fall of 2002, the instructor had to demonstrate how to open each software add-in during several consecutive classes. In addition, the functionalities of software toolbar buttons were described multiple times to account for students who were either absent, inattentive, or lost during the first few demonstrations. In the spring of 2004, when the multimedia modules were implemented for the first time, these questions rarely—if ever—arose. The class time gained by eliminating these repetitions was also followed by a decrease in the level of frustration experienced by the instructor at having to repeat software usage information so many times. Undoubtedly students could sense this frustration and eliminating it improved the atmosphere of the class.

In order to carry out the formal quantitative assessment, multiple-choice quiz questions were written for each multimedia tutorial developed. These questions were originally developed by OR faculty, then tested by math education faculty to ensure clarity and significance of the questions. The questions were designed to assess students' understanding of how to use the software, the purpose of the Excel add-ins, and recognition of the appropriate output resulting from specific software commands. The questions were not designed to assess students' mathematical modeling ability or understanding of modeling concepts. Questions from all four quizzes were combined to form a pre-test that was administered on the first day of class. Student responses were used to determine a baseline for comparison with the post-quiz responses. Multimedia tutorials were posted on the course website at least one week prior to the viewing deadline. On that date, the questions relating to the

given module were administered again as a post-quiz. In addition, as part of the post-quiz, students were required to supply a code word or phrase embedded in the module's audio track to ensure that they had actually viewed it. The post-quiz relating to a specific add-in or concept was administered prior to any class discussion on the same material. The students received full credit for simply completing the pre-test, while post-quizzes were graded for correctness. The pre-test together with the four post-quizzes represent five of six quizzes that students took throughout the semester (the sixth quiz was related to course content, not to the video modules). These six quizzes made up ten percent of the students' semester grade for the course.

Here are some examples of the quiz questions:

1. The Solver add-in to Excel is used to
   a. Maximize or minimize a linear function when known restrictions are present
   b. Compute the risk of several decision strategies relative to each other
   c. Solve a multivariate equation when all but one unknown is specified
   d. Evaluate decision strategies when uncertain outcomes are present
2. The end of a branch in PrecisionTree is represented by a
   a. Red circle
   b. Yellow diamond
   c. Blue triangle
   d. Green square
3. A two-way data table
   a. Computes the value of two outputs for any number of unknowns
   b. Computes the value of an output for two different unknowns
   c. Computes the value of an output for two different values of the same unknown
   d. All of the above

Interested readers can obtain the complete list of questions from the authors.

Twenty-five students enrolled in Mathematical Modeling in the spring of 2004. One student did not come to the first class and missed taking the pre-test. Four other students did not complete the post-quiz for one or more of the modules. Since the data set was incomplete for these five students, they were not included in the assessment but a complete set of pre-test and post-quiz data was collected and evaluated for twenty students.

The mean and standard deviation of correct responses for each module appears in Table 1. Based on the pre-test values, students began the semester answering approximately one question related to each module correctly. At the end of each module, the number of correct responses increased significantly.

Table 1 Pre-Test and Post-Test Data for Each Online Module

	Pre-test data		Post-quiz data	
Module	Mean	SD	Mean	SD
Data Tables (5 questions)	1.25	0.97	2.45	1.32
Solver (5 questions)	1.65	0.88	3.65	1.09
PrecisionTree (6 questions)	1.10	1.02	5.00	1.38
@RISK (6 questions)	1.80	0.89	4.55	1.15

This was formally determined with a t-test statistically comparing the mean of the pre-test correct responses with the mean of the post-quiz correct responses at the 1% level of significance. In each case, the mean of the post-quiz responses was significantly larger than the mean of the pre-test responses. Therefore, students completed each module with a better understanding of how to use the software featured in the module and the type of output to expect from the application of software commands. They also completed each module with insight into the purpose of the Excel add-ins. While representing a significant pre-test/post-quiz increase, the mean post-quiz value for the Data Tables module was lower than any other module. This was the first module and post-quiz the students took. Therefore, the lower post-quiz mean may be the result of the newness of the modules and student uncertainty about what the post-quiz would entail after they viewed the module.

After the Spring 2004 semester was completed, we asked for feedback about the video modules from the students who completed Mathematical Modeling. Since we made this request after the semester was completed, we received only four responses. Based on this small number, the students felt that the purpose of the tutorials—to help students become familiar with important Excel features and add-ins before they were needed to study a particular Mathematical Modeling topic—was achieved through the video modules. One student expressed frustration with the technical aspects of playing and rewinding the video, but the other three found the tutorials to be user-friendly and easy to understand. The students found the ability to simultaneously view the video and practice the skills covered by the tutorial in a separate spreadsheet window, pausing the tutorial when necessary, was a useful pedagogical approach for learning the skills covered by the modules. When asked if these modules should be used in future Mathematical Modeling classes, the respondents felt that they were a good introduction to Solver, PrecisionTree, and @RISK, as well as a useful way to learn about Excel data tables, and should be part of future classes. We believe that student feedback on whether these modules are achieving their purpose is important. Therefore, when these modules are viewed by

future classes, we plan to survey students for their opinions before the semester is over.

6. Development Issues

While we believe that the multimedia tutorials we developed were important additions to our course materials, we do not claim to have constructed perfect audiovisual supplements in our first attempt at this endeavor. Informal discussions with students and colleagues, as well as comments from a thoughtful reviewer, identified improvements that can be made to the existing modules. Perhaps the most difficult task in the process is establishing the proper flow of the modules and coordinating timing of all essential elements. For instance, some viewers found it confusing to listen to the audio voice-over while watching a visual demonstration of a software task, preferring instead to hear the explanation in full before the visual demonstration began. Furthermore, the most effective sequence for recording voice-over, slide timing, and screen capture is not clear. It is difficult to determine the pace of a visual demonstration without hearing the recorded voice-over, but recording the audio portion first allows for little flexibility in timing (if, for example, a particular task takes longer to demonstrate than anticipated).

Another significant issue that limits the quality of these tutorials is the clarity and continuity of the captured video images. It is sometimes difficult to read the menu options in the visual demonstrations, and the videos at times skip frames, making them seem very discontinuous. Similarly, we faced difficulty with volume consistency, especially when utilizing multiple audio files in the final presentation. While Microsoft Producer claims to normalize audio levels, we nonetheless found this issue difficult to circumvent. As a result, volume levels sometimes vary throughout a presentation. We suspect that these issues are a result of our software selection and would likely be avoided by purchasing a higher caliber tool designed specifically for building supplementary instructional modules.

In addition to the issues we faced in constructing the modules, we faced difficulties in delivering them. As noted earlier, students without adequate internet connections were unable to utilize these modules effectively from home. In fact, students with dial-up connections were not able to view them at all. This issue did not affect our use of the modules, however, as students had liberal access to a computer lab with machines identical to those on which the modules were built and tested. We would suggest that others without this resource consider making multimedia tutorials available on CD-ROM for those students with insufficient internet connections.

By far, the first tutorial took the longest to develop. Designing and recording the first tutorial involved thinking through many of the content and flow issues and learning the features of the production software. (We should note that Microsoft Producer is very user friendly, yet it may take some time to learn exactly what its capabilities are.) Once the first module was complete, however, the time required to record the remaining modules was significantly reduced. We do feel, however, that without writing scripts and establishing the "story" of the modules first, that development would have taken significantly longer.

7. Concluding Remarks

The formal assessment and student feedback reveal that online video tutorials are a useful way for students to practice basic spreadsheet skills before needing those skills to study mathematical modeling concepts. Requiring students to view the modules out of class and giving quizzes as an incentive to spend time practicing the skills allowed for valuable class time to focus on the fundamentals of the course—concept development, practice with model building, and analysis of solutions from spreadsheet models.

While the time investment required to develop these tutorials was nontrivial, we consider it well worth the effort. We hope that sharing our process serves as encouragement to those contemplating similar approaches. While our discussion has been limited to a few specific Excel add-ins, we see no reason that a similar approach could not extend to other quantitative software packages. In fact, multimedia tutorials that instruct in basic software functions could be used for almost any menu-driven software package, even those that are not quantitative.

Beyond basic software instruction, we can envision using similar multimedia modules to enhance learning in Mathematical Modeling and other operations research courses. For example, tutorials similar to those described in this paper could be used to walk students through development of homework solutions. Tutorials could be developed for more advanced undergraduate and graduate operations research courses to address more advanced software utilities and stand-alone software packages. Ultimately, such tools could be the beginning of the development of related distance learning courses for our institution.

Acknowledgments

We wish to thank Jason Merrick for his suggestions and assistance regarding the module development and assessment described in this paper. We also wish to thank the referees for thoughtful comments which improved the presentation of our work.

References

Albright, S. C., W. L. Winston, C. J. Zappe. 2001. Excel Tutorial to Improve Your Efficiency, http://www.indiana.edu/~mgtsci/files/ExcelTutorial.doc.

Belton, V., J. L. Scott. 1998. Independent learning and operational research in the classroom. *J. Oper. Res. Soc.* **49**(9) 899–910.

Belton, V., M. Elder, H. Thornbury. 1997. Early experiences of mentoring: Design and use of multimedia materials for teaching OR/MS. *Omega* **25**(6) 659–676.

Callahan, E. R., Jr., Jp. P. Shim, G. W. Oakley. 2000. Learning, information, and performance support (LIPS): A multimedia-aided approach. *Interfaces* **30**(2) 29–40.

Crowe, D., H. Zand. 2001. Computers and undergraduate mathematics 2: On the desktop. *Comput. Ed.* **37** 317–344.

Daku, B. L. F., K. D. Jeffrey. 2001. Development of an interactive CDROM-based tutorial for teaching MATLAB. *IEEE Trans. Ed.* **44**(2).

Gass, S., D. Hirshfeld, E. Wasil. 2000. Model world: The spreadsheeting of OR/MS. *Interfaces* **30**(5) 72–81.

Grossman, T. A., Jr. 2001. Causes of the decline of the business school management science course. *INFORMS Trans. Ed.* **1**(2), http://ite.pubs.informs.org/Vol1No2/Grossman/.

Liebman, J. S. 1994. New approaches in operations research education. *Internat. Trans. Oper. Res.* **1**(2) 189–196.

Palisade Corporation, Site Map, http://www.palisade.com/html/index.html.

Powell, S. G. 1995. The teachers' forum: Teaching the art of modeling to MBA students. *Interfaces* **25**(3) 88–94.

Powell, S. G. 2001. Teaching modeling in management science. *INFORMS Trans. Ed.* **1**(2) http://ite.pubs.informs.org/Vol1No2/Powell/.

Seal, K. C., Z. H. Przasnyski. 2003. Using technology to support pedagogy in an OR/MS course. *Interfaces* **33**(4) 27–40.

Smartforce, http://www.smartforce.com/learning_community/.

Troxell, D. S. 2002. Optimization software pitfalls: Raising awareness in the classroom. *INFORMS Trans. Ed.* **2**(2) http://ite.pubs.informs.org/Vol2No2/Troxell/.

Troxell, D. S., J. Aieta. 2000. Teach yourself solver. *Proc. NEDSI 2000*, Atlantic City, NJ.

Wahl, A. 1995. College students design software to explain tricky concepts. *The Chronicle of Higher Ed.* **42**(12) A20.

Wiest, L. R. 2001. Using information technology in mathematics education. *Comput. Schools* **17**(1/2) 41–55.

Winston, W. L., S. C. Albright. 2001. *Practical Management Science*. Duxbury, Pacific Grove, CA.

Wood, S. L. 1996. A new approach to interactive tutorial software for engineering education. *IEEE Trans. Ed.* **39**(3) 399–408.

* * *

To reference this paper, please use:

Hardin, J. R. and A. J. Ellington (2005), "Using Multimedia to Facilitate Software Instruction in an Introductory Modeling Course," *INFORMS Transactions on Education*, Vol. 5, No 2, http://archive.ite.journal.informs.org/Vol5No2/HardinEllington/.