Strategies for Teaching Computer Skills to First-Year Engineering Students

Larry G. Richards
University of Virginia

Abstract

What computer skills should freshman Engineering students master? How should they learn these skills? At the University of Virginia, a first year *Introduction to Engineering* course emphasizes spreadsheets (Excel), Computer Aided Design (SilverScreen), and a mathematics problem solving and symbolic manipulation program (MathCAD), as well as Internet and World Wide Web skills. A series of scripts (lab lessons) leads the students through selected capabilities of each program. Our approach stresses minimalist learning – the students learn the essential commands and capabilities of each program, and then elaborate these on their own. The scripts include some instructions that cause errors or lead to problems. This forces the students to think about the limitations of the software, and how to overcome them. They also learn that all software has flaws. Each lesson has an associated assignment to be completed outside the lab. Two major projects include developing a personal webpage, and a CAD design competition. This paper includes examples of student work and sample scripts.

1. Introduction

In their first year at the University of Virginia (UVa), our students are exposed to two computer laboratory experiences. The first semester they take *Introduction to Engineering* and, in the second semester, *Introduction to Computer Science.* The latter course introduces computer programming using C++; the former uses applications programs to solve problems in engineering and applied science.

Computing applications and engineering problem solving are taught in the lecture and lab sections of a first year, first semester *Introduction to Engineering* course. This course includes design workshops as well. The design workshops involve three team based design projects. As the semester progresses, the material from lectures and labs is integrated into the design projects.

Our goal is to develop our students' computing skills from the very start of their educational experience. The first lab introduces the computing environment at UVa, and covers basic use of the Internet and World Wide Web. In subsequent weeks, Excel, SilverScreen and MathCAD are each introduced with a couple of lessons and then periodically revisited as the term progresses – with elaboration and refinement each time. This pattern of interweaving lessons requires the students to review past lessons and to keep their skills in all programs current. Labs are offered five days a week for 1.5 hours every morning (Monday through Friday) in a dedicated computer facility that accommodates up to 120 students at a time. In each lab, a faculty member teaches the lesson with the assistance of a corps of graduate teaching assistants and one or two other faculty. Thus, any student having difficulty may get immediate help and personal attention.

Why did we choose these particular applications programs? They represent a diverse range of applications, and are useful in a variety of disciplines. They are readily available to our students; their vendors provided attractive arrangements for purchase of the programs by our students and affordable site licenses for the school. A faculty committee selected these particular products after considering a range of options for each type of software. Our goal was to introduce the major capabilities of each type of software. We also selected these products based on ease of learning and ease of use. Our committee surveyed the faculty in Engineering and Applied Science at UVa about the skills they felt our students should have; we also asked them which software they actually used in their classes, research, and consulting. Finally, we considered the demands and needs of industry for computer fluent engineers. We cannot provide instruction in each of the particular software products used by industry, but we can and do educate our students in the relevant capabilities and limitations of each type of program.

2. Incoming population: a profile of our students' computer skills

What do our students know when they arrive? We have surveyed our incoming students for over a decade about their computer skills and ownership. Most students now report having previously used MS Excel and MS Word. A few students have used a CAD system, usually AutoCad for drafting. None of our students have used MathCAD before, but almost all of them report frequent use of the Internet and World Wide Web. Less than half of our students report having good programming skills (this number has been going down in recent years). However, some claim to know at least two languages and a few half a dozen. Thus, a small number of incoming students are true experts with computers; a few have no computing experience at all; and most are familiar with some computer applications, but not very sophisticated in general.

In addition to differences in computing background and skills, our students also vary widely in learning styles and motivation. Some students become excited about what they are learning and try to master these programs to a high level of expertise. Others must be forced to achieve minimum competence. In labs, about 80% of each class feels the level of instruction is about right. Of the remainder, about as many object that the class is too difficult as say it is too easy. Regarding the lab scripts, some students resent being told too much, others feel we tell them too little; most work through the assignments with no complaints.

3. Philosophy and pedagogy; Design of the lessons

In lectures, we provide the motivation and context for using the computer programs, and introduce the necessary substantive background material. We may demonstrate techniques and principles using these programs, but we do not try to teach the software in lectures.

The lab is the fundamental learning environment; our students learn by doing. Each lab involves a lesson with examples, an in-class assignment and then a homework assignment to be completed during the next week. For some labs we require most of the period to lead the students step by step through the lesson; for most we provide a short introduction to the lesson, and then let the students work through it at their own pace.

The lessons are designed to be completed in the context of the labs. They are not intended for self-study (although some students use them that way). In the labs, personal help is always available. When students try to complete the assignments on their own outside the lab, they frequently make mistakes or take an inordinate amount of time figuring things out. When a student complains about how much time they spend on our assignments, we know they missed lab (or arrived late - after the preliminary remarks explaining the lesson).

Preparing the lessons is a major challenge. They are designed with awareness of the variety and range of individual differences between students. The labs are not geared toward the top of the class, nor the bottom. They cannot be too easy, or too complex. We know that most of our students have a moderate level of computer sophistication. Individual help is available for those who fall behind or do not understand the material. We have worked for years to get the level and timing of the lessons right; most students can complete most lessons with very little help. But each year we discover new problems: some due to the changing software, others due to the students. The lessons must be regularly updated to reflect software revisions and changed to incorporate new content and exercises.

The labs embody the philosophy of Minimalist Teaching (1,2): we teach a minimal set of commands for each program; we introduce just enough information to get our students started and enable them to accomplish something interesting and useful. We encourage them to experiment and to play with the software. We let the students make mistakes; indeed we force them to do so. In labs, we are prepared to rescue or assist them, but not too soon. Ideally the students will discover and correct errors on their own. Sometimes the labs contain unintentional mistakes. The labs are revised each year. No matter how careful we try to be, we always discover problems, ambiguities, or new bugs during the first session of a new lab.

4. The content of the lessons (what we teach)

Table 1 is the schedule of labs for Fall Semester, 2000. Both the particular capabilities and the pattern of interweaving the material are evident from the table. In addition to the weekly lessons and homework, we assign two major projects: a personal web page, and a CAD (SilverScreen) modeling project. These projects require creativity, planning, and timely implementation. They make our students go beyond what we teach them, master advanced capabilities of the software, and expand and integrate their knowledge. We want them to see these programs as creativity tools, not merely as ends in themselves.

In their first lab, our students learn to use the Internet and World Wide Web, including how to navigate and search the WWW; and what UVa resources are available on the web. The first assignment is a search for information about people, departments, labs, administrative offices, and facilities at UVa. The associated homework involves accessing other schools around the world, government agencies, professional societies, and selected corporate websites. Students are asked to reflect on the features of the sites they access, and note good and bad practice. What makes sites easy to use, and what causes confusion or frustration? They are told in this lab that they will have to design a personal web page, and asked to start planning its content and features.

Table 1. Engr162: Computer labs

Lab	Dates	Topics
1	Aug 30 - Sept 5	Computing at UVa, e-mail, home directory, Internet and WWW
2	Sept 6 – 12	Excel: basic features, plotting graphs
3	Sept 13 – 19	SilverScreen: basic 3D modeling
4	Sept 20 – 26	Excel: advanced graphing; functions
5	Sept 27 - Oct 3	Excel: statistics and linear regression
6	Oct 4 – 10	SilverScreen: Boolean Operations
7	Oct. 11-17	SilverScreen: additional modeling techniques
8	Oct 18 – 20	Web page design help sessions
	Oct. 20	**Personal Web page due**
9	Oct 25 – 31	MathCad: basic features and capabilities
10	Nov 1 – 7	MathCad: Matrices
11	Nov 8 – 14	Solving Equations with MathCad and Excel
12	Nov 15 – 21	SilverScreen: data structure
13	Nov 27 - Dec 1	MathCad: Calculus
	Dec. 1	**SilverScreen design project due**
14	Dec 4 – 8	MathCad: Advanced Topics

Many students have already used Excel. We cover the basics of spreadsheets and then those capabilities most useful in Engineering and Applied Science: plotting data, intrinsic and user defined functions; scales and transformations; descriptive statistics, curve fitting and linear regression, and matrices.

We teach Computer Aided Design as solids modeling. Even those students with previous CAD experience have typically not done true 3D; they have learned 2D drawing, generally using AutoCad. We use the SilverScreen solids modeling program to teach basic sweep techniques (linear, circular, ruled surface, and path), 3D and2D Boolean operations, parts libraries and assemblies, viewing and rendering, the data structure and annotation of a model.

MathCAD is new to all our students. We introduce the essential features of the user interface and the key toolbars, and then cover how MathCAD can be used in statistics, functions, matrices, calculus, and iterative techniques. Whenever appropriate, we compare the approaches of MathCAD and Excel to similar problems. Sometimes when they complete the same homework problems with Excel and MathCAD; our students get different answers. We ask them to reflect on their answers and explain any discrepancies. With matrices; MathCAD will not perform illegal computations, instead it provides instructive error messages; Excel just does the computations even when the result is wrong. In Excel, it is possible to add or multiply non-conformable matrices. We stress the importance of checking answers and how to do it. We take advantage of the limitations of the programs to get students to think about their solutions. Our students are learning what computers can’t do, as well as their capabilities, and we hope developing caution with respect to their results.

Example lab scripts are shown as Appendices. The lab scripts are the primary mechanisms for learning, but we also assign a text related to each program (3,4,5).

We require our students to comment on their results, to reflect on their meaning, reasonableness, or significance. The lessons sometimes contain incomplete instructions, embedded problems and mistakes; we intentionally ask students to do things we know will cause them problems. This upsets some of our students; they are disturbed by uncertainty and ambiguity. “Why don’t you use software without bugs?”

Another frequent complaint from our students is that we are not using some program they already know. They want to do the assignment in AutoCAD, QuattroPro, or Maple. Why should they have to extend any effort to get through this class? This is a freshman phenomenon. After the first year, those students who stick with Engineering are anxious to learn as much as they can about various computer programs. The more they can list on their resumes, the better.

In all our teaching, we encourage experimentation, discovery, troubleshooting, testing, and play. We want our students can learn more than we teach them. We hope to provide a good start that allows the students to progress to greater levels of sophistication on their own. We associate computer use with engineering problems from the start, and try to ensure articulation with other courses our students are taking (TCC, Math, design).

The two design projects (web page and CAD model) are presented as contests. The students compete with others in their design workshops and the rest of the class. Each design workshop has three entries into the class competition. Thus nearly 500 entries are reduced to 48 for further judging. The faculty selects the ten best designs and then the entire teaching team (faculty and graduate teaching assistants) vote on the final order of these designs. Each year the quality of the entries increases, as the students see what their predecessors have done, and try to top it.

5. Beyond the first year

Once students have these computing skills, how can we ensure their continued use and elaboration? The purpose of teaching these skills in the first year is so they can be used, refined and developed throughout the curriculum. However, current reality does not yet match this ideal. Except for Computer Science majors, few students will be required to program a computer in the rest of their undergraduate curriculum. Many will be able to use the spreadsheet, CAD, and math programs in their classes, but few will be required to, and some will be discouraged from doing so.

There are some exceptions. Second year Mechanical Engineering students learn AutoCAD, and in their third or fourth years may learn SolidWorks, Pro Engineer, and/or SDRC Ideas. Students in Civil or Electrical Engineering will cover different CAD programs. In Chemical Engineering and Systems Engineering, simulation methods are required. A popular elective course on *Computer Graphics* allows some students to master graphics programming in Visual Basic. Mechanical and Electrical majors can also learn MatLab and LabView. Some classes include finite element analysis, mechanical dynamics, or computational fluid dynamics.

6. Computer skills minor

Students should be able to pursue a minor in computer skills. This would help establish their expertise with computer applications, and add to the market value of their degrees. UVa already offers a minor in computer science but a minor in computer applications would make more sense for engineering students. The former includes Introduction to Computer Science, Discrete Mathematics, Software Development Methods, Program and Data Representation and two CS electives**.** The latter could include Computer Graphics, simulation methods, CAD/CAM, finite element methods, computational fluid dynamics, numerical methods, and discipline-specific courses such as computer applications for aerospace engineers. These courses could be structured around real engineering problems and involve teams of students in major projects. Whereas all the courses in the CS minor are taught by CS faculty, most of the computer applications courses would be taught by faculty from the engineering departments.

7. Conclusions

Over the years, we have received positive reactions from our students about this course, especially regarding the computer labs and design workshops. They are less enthusiastic about lectures. We have also had some positive feedback from other faculty. Several design instructors require CAD models and spreadsheet analyses for each team project. In their second semester, our students all take a required Physics course. After we introduced MathCAD into our *Introduction to Engineering*, our students were able to do more complex problems in Physics. The Physics instructor was so impressed that he is revising the Physics course to take advantage of these new skills. He will incorporate more complex and realistic problems in the new version of his course. One second-year

Dynamics instructor was amazed and delighted at the professional results and presentations his students achieved using Excel and MathCAD.

The first year of engineering education is clearly the best time to immerse students in computing. The skills they acquire should be used and further developed throughout their engineering curriculum. J.B. Jones (6) asserts that this is often not the case. Computing is not well integrated into most engineering curricula. However, ABET requires that students be able to handle the tools of modern engineering practice; members of our Industrial Advisory Boards tell us that computer skills are essential; and our graduating students with the most extensive computing backgrounds seem to get more and better job offers. These messages are not lost on our students. The faculty will eventually have to adjust their thinking, and teaching, to accommodate these new realities.

Bibliography

1. Carroll, J.M. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*. The MIT Press, 1990.

2. Carroll, J.M. The adventure of getting to know a computer. *IEEE Computer*, 15 (11), 49-58, 1982.

3. Gottfried, Byron. *Spreadsheet Tools for Engineers, Excel 2000 Version*. McGraw Hill, 2000.

4. Faldowski,J., Colledge, T., Sathianathan, D., Ranade, S.,and Meyer, K. *Computer Aided Design Using Solid Modeling, Seventh Edition*, Schroff Development Corporation 2000.

5. Larsen, Ronald *Introduction to MathCAD2000*. Prentice Hall, 2000

6. Jones, J. B. “The Non-Use of Computers in Undergraduate Engineering Science Courses” *Journal of Engineering Education*, January 1998, Vol. 87, No.1, 11-14.

LARRY G. RICHARDS is an Associate Professor in the Department of Mechanical and Aerospace Engineering at the University of Virginia. He is Director of the UVa Master’s Program in Manufacturing Systems Engineering, and of the Center for Computer Aided Engineering. Since 1992, he has taught computing fundamentals to first year engineering students. His current interests include Engineering Entrepreneurship and Invention and Design. He may be contacted at lgr@virginia.edu.

Appendix 1

ENGR 162 – Excel Lab # 2

Graphs with Different Types of Axes

Create a 4-column, 10-row table starting in cell B4. Enter the values of x from 1 through 10 in cells B4..B13. Let y = 7*x + 5 and enter the ten corresponding values of y in cells C4..C13. Enter the values of x-squared (x^2) in cells D4..D13. Enter the values of exp(x/2) in cells E4..E13.

When the mouse cursor is pointing at any object, clicking the right mouse button activates a menu that in most cases allows you to change the object's properties (as well as cut, copy, and paste). You will need to do this in performing some of the following tasks:

* Enter labels for each column in the appropriate cells in row 2.

* Center the cell contents in each column, both labels and values by using the formatting toolbar.

* Create an XY graph with the three series of values for y, x-squared, and exp(x/2). Note that only one of the x vs y series produces a straight line!

* Copy the original graph to another location on the spreadsheet and change the scale of the Y-axis to a logarithmic scale. Note which series is a straight line now!

* Copy the graph again and change the X-axis to a logarithmic scale so that you now have a log-log plot. What happens in this case and why?

* Place appropriate titles on each of your graphs.

* Save your final spreadsheet and the three graphs on your floppy diskette, or Home Directory.

Curve Fitting (by Linear Regression)

In engineering work, the need often arises to "fit" a line or curve to a set of experimental data points. Sometimes, enough is known about the physical situation that the engineer can assume the curve has a particular form, such as a straight line. Excel has the capability to fit a straight line to a set of data points by calculating the slope and Y-intercept of the straight line that makes the best fit. It also calculates information on the closeness of the fit. The process is called **linear regression.**

Make a spreadsheet table with two columns containing the following values:

1	10
2	20
3	30
4	30
5	40
6	50
7	50
8	60
9	70
10	80

The first column contains values of the independent variable. The second column contains values of the experimental, dependent variable. **Be careful, you cannot speedfill column 2.**

* Now select the Tools menu, then the Data Analysis sub-menu, and then the Regression command. *(On your own computer, the Analysis ToolPak may not be installed. If you do not see Data Analysis under the Tools menu, select Add-Ins, and enable the Analysis ToolPak.)*

* Specify the block containing the dependent variable values (Y range), the block containing the independent variable values (X range), and under output options, select output range and select a cell marking the upper-left corner of the output block (where the output will be printed on the spreadsheet). Also check the residuals box.

* Press the OK button and watch what happens. Find the slope (X Variable) and the Y-Intercept (Intercept). Look at the information given and see if you can identify what it is.

* Use the experimental values to draw a graph consisting of points only with no connecting lines.

* Use the calculated values of the Y-Intercept and the slope to plot a straight line over the experimental data points already plotted. To do this, create a third column of values next to the already existing two in your spreadsheet table (use the absolute address command F4 in doing this). Again, use the right mouse button to go about deleting the point markers from this straight-line plot.

* Save your spreadsheet and results.

In lecture Professor Cahen has shown how linear regression can be used to fit equations to data in linear, semi-log, and log-log coordinates. He also discussed which types of equations plot as straight lines in each of these systems. You will need to understand these concepts and techniques for this week's homework. These ideas are covered in Gottfried's chapters 3 and 5.

Assignment (to be turned in at the beginning of lab one week from today.)

From Byron Gottfried's *Spreadsheet Tools for Engineers 2000 Version*, please do the following problems and answer the associated questions (both ours and Gottfried's). **All these problems are to be completed using Excel. All graphs should be appropriately labeled and captioned. Any questions should be answered on the EXCEL spreadsheet. Remember to enter your name, student number, workshop number and instructor on every sheet (as part of the computer output, NOT added by hand.) Format your output so that each problem is on a single page. Staple all the pages and remember to pledge your assignment.** See how many of these problems you can complete this morning while help is readily available.

Exercise 1. Page 74, problem 3.9. Is a straight line a good model for this data? How good do you think the fit will be?

Exercise 2. Page 78, problem 3.11

Exercise 3. Page 84, problem 3.14

Exercise 4. Page 124, problem 5.5. Be sure to show a graph with the best fitting line as well as the data points. On your spreadsheet, write out the equation for the best fitting straight line for this data. How good is the fit? How do you know?

Exercise 5. For problem 3.17 on pages 85 and 86 of Gottfried, use linear regression techniques (as described in Prof. Cahen's lecture) to find the actual equation which best fits this data and plot it in Cartesian coordinates. First, find and plot the best fitting regression line in the appropriate coordinate system. Once you know the relevant values for the coefficients, use the appropriate transformations to find them for the equation in Cartesian coordinates. Show both the data and the equation in the graph. Be sure to show the equation as well as its plot.

Appendix 2

SilverScreen Lesson 2

BOOLEAN Operations on Models (3D objects).

1. Using DRAW - SOLIDS, place a sphere and a cylinder on your screen. Position them so that the cylinder passes completely through the sphere, and the center axis of the cylinder passes through the center of the sphere. Measure the volume of each of these objects (using GEOMETRY-MEASURE-OBJECT) and record the results in an Excel spreadsheet. Obtain the UNION of these two models using

```
                    GEOMETRY
                      BOOLEAN - 3D
                        UNION
```

Pick a primary object and then a secondary object. Watch the action on the screen as SilverScreen combines them. Visualize the result by generating a four-way window split and then shading the oblique view (select Render - Hidden surface removal). Print out the result. Remember to include your name, etc on the printout.

2. Repeat this exercise, but take the DIFFERENCE between the objects. First, subtract the Cylinder from the Sphere. Examine the resulting solid. Then select EDIT-UNDO, and subtract the Sphere from the Cylinder. The difference operation yields quite different models depending on which solid is considered the primary object and which the secondary object. In each case, measure the volume of the resulting object, and record it in your Excel spreadsheet. Print out a copy of the first construction (sphere minus cylinder) with hidden lines removed in the oblique view

3. Finally take the intersection of these two SOLIDS. Measure the volume of the resulting object, and record the result.

4. Enhance the appearance of your spreadsheet- add labels, center things, use consistent number formats, maybe some lines, and certainly your identification file.

Exercise 1 is now complete. You should turn in the two printouts of models, and the spreadsheet summarizing the volume measures with this week's homework. Does the pattern of results make sense? What are the units for these measures?

Making complex shapes from simple ones: 2D Boolean Operations

1. Draw two overlapping rectangles on the screen. Arrange them in the shape of a plus sign (+). Select

```
                    GEOMETRY
                       BOOLEAN - 2D
                          UNION
```

Identify one rectangle as the primary polygon, then the other as the secondary one. Select each by clicking on the appropriate object. Watch the screen to see how SilverScreen obtains and displays the UNION of these two shapes.

2. Draw two more overlapping rectangles and obtain their intersection using

```
                   GEOMETRY
                      BOOLEAN - 2D
                        INTERSECTION
```

Again a primary and secondary polygon must be selected. With UNION and INTERSECTION, the selection order is not important. With DIFFERENCE, the order of selection is critical.

3. Draw two overlapping circles on your screen. Then select:

```
                     GEOMETRY
```

BOOLEAN - 2D

DIFFERENCE

Choose one shape as the primary form, and subtract the secondary one from it. Undo the result, and reverse the selection order.

4. Now clear your screen (FILE - CLEAR ALL - YES) and draw a circle, triangle, and rectangle on the screen and experiment with these BOOLEAN commands (UNION, DIFFERENCE, INTERSECTION). Try to envision each result before you generate it.

Fillets and Chamfers

1. Draw again the two overlapping rectangles in the shape of a plus sign. Now select the following sequence of commands:

```
EDIT
    FILLET
        (Select a polygon)
            INSERT ALL
                (Enter fillet radius)
```

Repeat for the second rectangle. Take the UNION of the two resulting figures.

2. Draw another pair of overlapping rectangles. This time take their UNION first, and then FILLET all corners in the resulting figure. Note the difference in the result from the previous construction.

3. Place a rectangle on the screen. Select:

```
EDIT
    CHAMFER
        (Select the polygon)
            INSERT Single
                Click on a corner
                    (Enter 2 lengths)
```

The Grid and Snap Interval

For accurate drawing, we can draw to a grid and have points snap to the nearest grid locations. To set the grid, select:

```
DRAW
    TOOLS
        GRID
```

First enable the grid, and then change the grid spacing from 1.000 to .250. Next, change the grid pattern to lines. Finally try a different line color.

To change the SNAP interval, select:

```
DRAW
    TOOLS
        SNAP
```

Change the SNAP interval to .25, enable the snap and link it to the grid.

Practice drawing to the grid. Construct a fairly complex COMMON POLYGON. Think about the lengths of the sides as you draw them. With a little practice, you will be able to draw metrically precise figures quickly and easily.

Assignments (Complete all four exercises to hand in at the start of lab next week)

2. Using 2D Boolean operations, create the profile shown in Chapter Exercise 1 on page 3.19 of your SilverScreen text using only rectangles and circles as primitive shapes. You may also use fillet or chamfer commands in creating this profile. Using linear sweep, convert your profile to a 3D model. Obtain a four-way split screen. Place your name on the drawing (use ANNOTATE - INSERT TEXT) and print it out.

3. Next, create a block and two cylinders: one cylinder should pass completely through your block, and the other only half way through it. Both cylinders should start above the top surface of the block. Subtract each of the cylinders from the block using 3D Boolean operations. Generate a four-way split screen drawing of this model, and render the views appropriately; we should be able to see the effects of both cylinders on the final model. Print out the result.

4. Drawing to a Grid: In the SilverScreen text on page 3-5 to 3-7, Faldowski et al describe creating a profile as a series of lines. The coordinates of the points are given near the bottom of the page (starting with 2,2,0). Create this profile by constructing a Common Polygon and drawing to a grid. Set the grid parameters so they are appropriate to this problem. Then, add the chamfers and fillets described on pages 3-6 and 3-7. Don’t follow the directions in the book. Think about what you are doing, and develop easier ways to accomplish the same results. Linearly sweep this profile, and printout the final result.