Increasing Time Spent on Course Objectives by USING COMPUTER PROGRAMMING TO TEACH NUMERICAL METHODS

DAVID L. SILVERSTEIN
University of Kentucky • Paducah, KY 42001

The chemical engineering curriculum is a crowded agenda, with students needing to allocate limited time resources to focus on educational objectives. A typical curriculum includes instruction in numerical methods, and programming assignments are one way of ensuring that students understand the methods underlying modeling calculations. Developing applications from "scratch" requires too much time spent on tasks not central to course objectives, however, leaving too little time for the engineering problem. An innovative approach to this problem, "Template-Based Programming," minimizes time spent on the elements of programming outside of course and assignment objectives.[2]

Template-based programming (not to be confused with the software engineering term "template") provides the student with a fully functional application, in all respects but one—the subroutine containing models and numerical methods is deliberately empty, with variables necessary to communicate with the user exposed and well defined for the student. The student is required to write only the codes necessary to implement the model and appropriate numerical methods. This enables students to focus on the assignment objectives without unnecessarily concentrating on the use of syntax and structures not germane to the engineering assignment.

BACKGROUND

Newly graduated chemical engineers will potentially be called upon to perform a wide range of tasks involving computers. They will need to use current technologies requiring little programming skill, but they may also be called on to develop models and simulations that require some ability to develop procedural computer code. At times this will involve legacy codes, often in FORTRAN, dating back several decades. Sometimes this will require working in modern environments, such as writing complex procedural scripts in MATLAB, or integrating spreadsheet calculations with procedures written in Visual Basic.

Template-based programming provides a means for incorporating computer programming into courses while minimizing time the students spend on programming tasks unrelated to the course objectives.

The limited space available in the curriculum for computer training must be used to provide as broad a base as possible to enable graduates to adapt to the specific computing-related requirements of their employer as rapidly as possible. At the same time, it must be recognized that most chemical engineers are not expected to be applications developers,

David L. Silverstein *is currently Assistant Professor of Chemical Engineering at the University of Kentucky, Paducah. His special interests include learning-style centric educational software development, pedagogy-driven design of distance-learning classrooms, improving retention of mathematics for engineering courses, and cooperative experiences in process control between engineering and technology students. He holds a PhD and MS in Chemical Engineering from Vanderbilt University, and a BSChE from the University of Alabama.*

writing complete user-friendly programs from "scratch."

To maximize the value of each course in the curriculum, the integration of programming must be directed toward meeting educational and program objectives as defined by the individual department or program. Accreditation Board for Engineering and Technology (ABET) accredited curricula require that students learn "appropriate modern...computing techniques."[2] In most cases, chemical engineering programs have addressed that criterion in part by incorporating a programming course.[3] The most common language used is FORTRAN, with C, C++, and Visual Basic taught in many other programs.[3,4] Teaching procedural elements of scripting languages, such as those in MATLAB, Maple, and Mathematica, is proving to be an increasingly popular option among chemical engineering programs.[4]

The decision on how to incorporate programming into the curriculum should take into consideration why programming is important to chemical engineers. A CACHE Corporation survey[5] provides some interesting insight. Of the practicing chemical engineers surveyed, 92% never use FORTRAN or another computer language in their work. Furthermore, 86% of employers did not expect literacy in different computer language paradigms. Faculty members revising chemical engineering curricula to meet the needs of industry will find it challenging to determine the computing skills that should be taught as part of a core curriculum.[6] When the requirements of the ABET Engineering Criteria 2000 (EC2000) are also considered, the need to specifically address programming and computer use in the curriculum is even more pronounced.

Since there appears to be little need for practicing engineers to program, we must ask why we teach programming in most chemical engineering curricula. The most likely answer seems to be, because developing a program requires that the program author break down a complex problem into a logical series of steps in a syntactically rigorous language, with a flow of logic that should reflect the mental discipline an engineer should be capable of to solve challenging problems. Programming concepts are expected to strengthen two key facets of engineering education—problem formulation and problem solving,[7] although some studies indicate this may not be the case.[8] It can also be argued that programming languages are "a novel formal medium for expressing ideas about methodology,"[9] and thereby strengthen the communications component of the curriculum.

General purpose mathematical software (such as MATLAB, Maple, Mathematica, or MathCad) has been adopted by many chemical engineering programs with the intent of teaching students problem-solving skills requiring computer usage. Typically, this requires some degree of programming. The benefit of this approach is that the "overhead" of applications development is handled by the host application. One difficulty with this approach is teaching numerical methods with these packages when the program already has most numerical methods incorporated directly into the software. In this case, use of a high-level programming language may be more appropriate. In all cases, the opportunity for sufficient instruction of students in the software or programming language they are expected to use is critical.

Applications development requires that all details regarding a program be addressed, including memory management, user interfaces, buffer protection, graphics, file management, and operating system integration. Engineering students should be focused on the problem, addressing numerical methods, model accuracy and assumptions, stability, limits of applicability, communicating the model and results, and integrating the results with reality. The method described herein enables focus on the topics important to

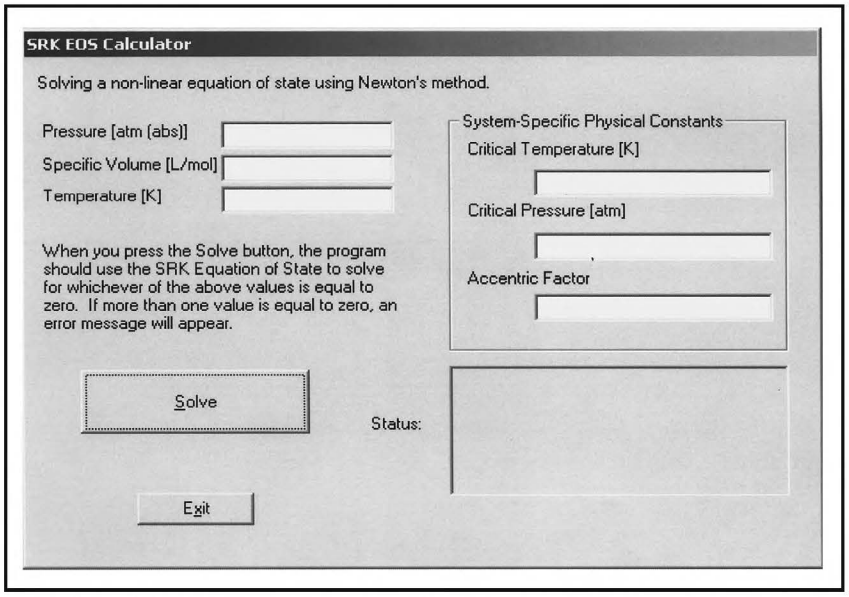

***Figure 1**. The user interface resulting from compiling and executing the template. The program is fully functional, except that it does not calculate anything when the button is clicked. Error messages regarding user input are still displayed, even without the student code.*

chemical engineering students.

TEMPLATE-BASED PROGRAMMING

The template-based approach begins with a fully functional application. The student is provided with the complete workspace for the integrated development environment (IDE) under which they have previously worked in their programming course. Immediately after loading the template, they can compile and run the program, which will appear as in Figure 1. One key element is, however, missing. When the "solve" button is pressed, nothing is returned. That button calls a subroutine that, while fully commented and linked to the application, is devoid of executable code. The student is expected to write all necessary code to provide the subroutine with the functionality required by the assignment. A typical subroutine is shown in Figure 2 . All variables needed as input for the routine are exposed by global variables, as is the variable declared to store the result calculated by the students, which is returned to the application interface.

The result of the assignment is an attractive, uniform, fully functional, stand-alone application that students can then use for solving additional problems. Time spent on non-objective activities, such as preparing "FORMAT" statements or reading and writing to files has been minimized or eliminated. The high-level programming language has been used in such a way that it shares many of the benefits of math packages, including the ability to assume that I/O management, memory management, and the user interface are taken care of without need for consideration by the subroutine programmer.

IMPLEMENTATION

Use of these templates develops along the same lines as other course material. First, expected outcomes for the assignment were derived from the course objectives. An assignment was developed and application constructed sans components required for the student to develop to achieve the expected outcomes.

Students were required to take a FORTRAN course to fulfill programming requirements until a policy change at the University of Kentucky (UK) in 2001 allowed them to take a course in Visual Basic to satisfy that requirement. To maximize the value of the student's previous training, early uses of this approach were coded in Compaq's Visual FORTRAN environment. Most recent uses of this approach have been assigned with a student option of using FORTRAN or Microsoft Visual Basic.NET templates. In all cases, students have been given the option of writing their own program from scratch, using any programming environment. The assignment is typically rather detailed, since some significant instruction is required to get started using the template because students have never had to work within someone else's application prior to this assignment.

RESULTS AND DISCUSSION

Students did not write routines to collect model parameters from a console or data file, design structures to pass data between routines, or format numerical output and send it to the console or to a file. They gained no experience in inheritance, encapsulation, polymorphism, or GUI programming. They did design a model, specify required input and output, write subroutines and functions involving the model and a numerical method, debug their code, validate the model, and verify the numerical method. They focused, thereby, on the course objectives—modeling chemical processes or performing engineering calculations, selecting and implementing numerical methods, and practicing fundamental logic and problem-solving skills.

An additional benefit of the approach includes ease of grading. Since all students started from a common set of variables, the code was easier to trace. The results presented in the form of screen captures were consistent and easier to follow.

Template-based programming has been applied in two courses so far: Process Principles, the sophomore material and energy balance course, and Process Modeling, a junior-level course in modeling principles and numerical methods.

The class size in all courses in which this approach has

```
SUBROUTINE SRK()
!Routine to actually run the SRK EOS calculations
!
! It obtains values of 3 of 4 variables (PVT) and Tc,Pc,and the accentric factor
!from the dialog box
! It must provide a value of the missing variable while not changing the variables
!provided
! Minimal error checking is performed
! The following two statements incorporate this file into the overall project
use dfwin
use SRKGlobals
!*******************************************************************************
!The following global variables are available to you.
!double precision     Pressure        Pressure given in the dialog box
!double precision     Volume  Specific Volume given in the dialog box
!double precision     Temperature     Temperature given in the dialog box
!double precision     TCrit   Critical Temperature given in the dialog box
!double precision     PCrit   Critical Pressure given in the dialog box
!double precision     Omega   Accentric Factor given in the dialog box (may be =0)
!You must assign a value to whichever variable from amongst  Pressure, Volume,
!Temperature was =0

!You will need to create some variables for use within your subroutine

!You may choose to create another subroutine INSIDE THIS FILE to handle equation
!solving

!You will need to first figure out what to solve for,and then branch to actually
!solve for that unknown
!That gives you three cases,one of which is trivial. For the other two, you will
!need to use Newton's method
!Remember to calculate the SRK parameters which you will use in all three cases.
!You will likely need to declare other variables to complete your program

!YOUR CODE STARTS HERE



!YOUR CODE ENDS HERE
END SUBROUTINE SRK
```

Figure 2. *Typical student subroutine template.*

been applied has been ten or less. The UK engineering programs in Paducah are an extension of the main campus in Lexington. Students pursue a BS in chemical engineering under what is currently a curriculum identical to that of the Lexington program. Chemical engineering courses at UK Paducah are taught by UK faculty stationed in Paducah, with the remaining courses taught by Murray State University and Paducah Community College.[10] ABET accreditation is expected in July 2003.

The first use of template-based programming was in Process Modeling. One of the objectives for this course is that students "use computer software and programming languages to solve complex mathematical systems." Two assignments were given. The first was a simple determination of the machine epsilon, the smallest difference between two numbers that the computer can distinguish. The purposes of the assignment were to reinforce the idea of inherent limitations of computer calculations, to reintroduce computer programming to the students (who had not programmed during the 1 - 2 years since their programming course), and to familiarize the students with the template-based approach. The second project was an implementation of Gauss-Siedel elimination to solve for a solution to a system of linear equations.

The first time this course was taught at the Paducah campus, students were unable to complete the epsilon calculation due to difficulties with passing variables to subroutines, with using DO loops, with reading and writing input and output, and other serious programming deficiencies. The options left to the instructor were to reteach these topics every time the modeling course was offered, to eliminate the programming task, or to develop a method that would allow students to focus on the skills they needed to retain.

The second time the course was taught, templates were used. Students were surveyed regarding their perception of their programming skills and the utility of the template-based approach.[1] Table 1 summarizes the results. Students demonstrated improvement in perception of their individual skills in various programming concepts after the template-based assignment in all cases except for the question about basic mathematical operations and on the question regarding DO loops. The second project required a sequence of nested DO loops that tested their understanding of this construct, which was not as strong as they had previously perceived. Other results of the survey indicated more time was spent on engineering objectives than would have been spent without the template.

The students were enthusiastic about this approach, stating that they were more confident about starting to program since the "busy work" had been completed for them. The visual nature of the output was more satisfying than the console-based output of programs they had written previously. Students from the previous section of the course expressed envy that students using templates did not have to struggle through the issues they did with fundamentals of applications development.

TABLE 1
Summary of Survey Results
("1" - student strongly disagrees: "5" - strongly agrees)
(Courses: CME200-Process Principles; CME420-Process Modeling)

	Initial Survey CME200	*Post Project Survey CME200*	*Initial Survey CME420*	*Post Project Survey CME420*
I am comfortable writing programs using FORTRAN	2.8	3.0	1.8	2.5
I understand how to use variable arrays in FORTRAN	3.6	3.9	1.3	2
I understand how to use if-then-else structures in programs	2.7	3.4	2.8	3.5
I understand the use of DO loops and how they are terminated	3.6	3.8	3.3	3
I understand the use of comparison operators in FORTRAN	3.3	3.5	3.3	4
I can perform mathematical operations on variables in a FORTRAN program	3.3	3.8	3.3	3.3
Using a template allows me to focus more on the engineering problem compared to writing a program from a blank file	N/A	4.5	N/A	3.8
The time spent programming was reduced by using the template	N/A	4.3	N/A	4.3
After having worked through the issues involved in starting to use the template, the template was easier to use than writing a program from a blank file	N/A	4.1	N/A	4.5

The biggest improvements observed by the instructor were the reduction in time spent during office hours instructing students in programming fundamentals and the fact that all students turned in programs that met instructional objectives. Students spent far less time working on the programs using the templates than before, but the time they did spend was quality time, focused on the assignment objectives.

The second course in which this technique was applied was Process Principles in the fall of 2001 and 2002. During coverage of equations of state, students were required to solve the Soave-Redlich-Kwong equation of state for an unknown PVT variable using Newton's method. Again, stu-

dents had issues with the numerical method, but not with the programming technique. In part, this was due to the freshness of the programming course in these sophomore's experience, but students still expressed a preference for working from a template instead of writing a program from a blank file. Submitted results were of superior quality compared to those of an assignment in the fall of 1999 that required complete development of an application to solve a similar problem.

Surveys were again conducted involving students in both sections of Process Principles in which templates were implemented. Table 1 summarizes those survey results. The key finding of the survey is confirmation that students again perceived that they spent more time focused on meeting course expected outcomes while practicing and improving their programming skills.

The template-based approach is flexible, readily applied not only to the Compaq Visual FORTRAN IDE and the Microsoft Visual Basic.NET environment, but applicable within MATLAB and other packages. In the fall of 2002, the Process Principles class had a group of students, half of whom had instruction in FORTRAN, and the other half a course in Visual Basic. Templates for both languages were made available and were used by students. The complete templates, including the code for the user interface, also serve as an example for students interested in developing their own applications for research or personal use.

There are some difficulties associated with the use of templates. The primary issue in implementing them is actually developing a reliable, debugged interface. With Compaq Visual FORTRAN, the interface development requires understanding of Microsoft Windows GUI development concepts, which is not part of the typical chemical engineering professor's background. More recent FORTRAN environments (Leahy Fortran.NET, for example) make this process much simpler. When students work under different languages or environments, multiple templates may be required, as they were during the fall 2002 offering.

The implementation of templates will benefit from some technical improvements. The subroutine should be encapsulated as an object, so that the student does not directly modify any variables from the host routines. Another alternative would be having the students develop the core of a dynamic link library that compiles independently from the application and may be called from a variety of other programs. Process simulators and data acquisition systems can also be integrated with the templates to further develop the student's computing skills.

To simplify adoption of this approach, the author has made templates freely available for download.[11] Other faculty who wish to share their templates with the engineering community at large are invited to submit their templates to the author for inclusion in a web-accessible library.

CONCLUSION

Template-based programming provides a means for incorporating computer programming into courses while minimizing time the students spend on programming tasks unrelated to the course objectives. It allows the student to model processes and apply numerical methods in a logical manner, breaking down complex procedures into syntactically rigorous steps, resulting in a usable application. It also capitalizes on the programming courses students take early in the curriculum.

Templates have been successfully incorporated into two courses at the University of Kentucky's extended campus in Paducah, with very positive results, particularly with respect to time spent on engineering tasks as described in the course objectives. It is a flexible approach, as demonstrated by use with both FORTRAN and Visual Basic, and can be applied to many other languages or computing environments. Stable and robust template development does require substantial time, but results in a more efficient educational experience for the student.

REFERENCES

1. Silverstein, D.L., "Template-Based Programming in Chemical Engineering Courses," *Proceedings of the 2001 ASEE Annual Conference & Exposition,* American Society for Engineering Education (2001)
2. *Criteria for Accrediting Engineering Programs,* Accreditation Board for Engineering and Technology, Inc., Baltimore, MD: <http://www.abet. org/> (2002)
3. <http://www.che.utexas.edu/cache/survey.html> CACHE Survey Results
4. Dahm, K.D., R.P. Hesketh, and M.J. Savelski, "Is Process Simulation Used Effectively in ChE Courses?" *Chem. Eng. Ed.,* **36**(3), 192 (2002)
5. Davis, J., G. Blau, and G.V. Reklaitis, "Computers in Undergraduate Chemical Engineering Education: A Perspective on Training and Applications," Technical report, CACHE Corporation, Draft 3.1 (1993)
6. Kantor, T.J., and T.F. Edgar, "Computing Skills in the Chemical Engineering Curriculum," in *Computers in Chemical Engineering Education,* B. Carnahan, ed., CACHE Corp, Austin, TX (1996)
7. Stephanopoulos, G., and C. Han, "Languages and Programming Paradigms," in *Computers in Chemical Engineering Education,* B. Carnahan, ed., CACHE Corp, Austin, TX (1996)
8. Urban-Lurain, M., and D.J. Weinshank, "Do Non-Computer Science Students Need to Program?", *J. Eng. Ed.,* **88,** 535 (2001)
9. Abelson, H., and G. Sussman, *Structure and Interpretation of Computer Programs,* MIT Press, Cambridge, MA (1985)
10. Smart, J.L., W. Murphy, G.T. Lineberry, and B. Lykins, "Development of an Extended Campus Chemical Engineering Program, *Proceedings of the 2000 ASEE Annual Conference & Exposition,* American Society for Engineering Education (2000)
11. <http://www.engr.uky.edu/~silverdl/TBP/> ❐