Template Based Programming in Chemical Engineering Courses

David L. Silverstein
University of Kentucky

Abstract

Chemical engineers today are called upon to do more with computers than their predecessors even a few years ago. Not only do they have to do more with new (and unfamiliar) technologies, they are also called upon to work with old (and unfamiliar) software that dates back 20 years or more. The limited space available in the curriculum for computer training must be utilized 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. When combined with a large projected shortfall in qualified personnel to fill computer related jobs over the next ten years, chemical engineers may find themselves required to fulfill some computer based duties previously delegated to MIS and other support personnel.

One approach to preparing students for the wide variety of computer related issues that may arise in their engineering duties is to take a template-based approach to developing engineering software. This method provides students with a software infrastructure, or template, that handles basic tasks, such as input and output, and requires them to utilize the basic programming skills learned in an introduction to programming course to develop the code to implement engineering designs and calculations. By using this method, the student does not have to learn, for example, how to incorporate dialog boxes in a GUI application, but focuses on the engineering aspects of the program.

Students will broaden their understanding of the software they use, as they are given the opportunity to examine code in the context of a program they understand better than an abstract calculational tool such as a spreadsheet or CAS. They will have written the "guts": only the "shell" is provided. Motivated students will work with the "shell" and deepen their understanding of broader programming concepts. The use of this approach in an undergraduate course in process modeling is examined. Some aspects of a proposed elective based upon this principle to broaden student exposure to other programming techniques (event driven, real time, etc. as opposed to procedural), languages (Java, C/C++, Visual Basic...), environments (Windows, UNIX, CE), object linking (custom ChemCAD modules, MS Office integration, DLLs), and interfaces (GUI, console, web) are presented.

I. Introduction

There has long been a debate over the role of computer programming in chemical engineering education. Programming in some form, whether it be a high-level language such as FORTRAN or C, or scripting within a mathematics application such as Mathworks MATLAB[1] or Wolfram Mathematica[2], is taught in the vast majority of chemical engineering programs in the U.S. The greatest challenge in incorporating programming into a chemical engineering curriculum is determining a method of utilizing computer programming to develop engineering skills without spending inordinate effort in tasks unrelated to chemical engineering.

At the University of Kentucky College of Engineering Extended Campus Programs in Paducah[3,4], a trial is currently underway to develop engineering skills by more effectively utilizing the training students currently receive in FORTRAN programming while minimizing time spent in what is arguably the most time consuming task in simple programming projects, designing input and output.

The premise behind the method is that students tend to spend significant amounts of time developing code to handle basic operations in an application, things like reading formatted data from a file, writing console input routines, and then presenting output in a readable form, either to a data file, a printer, or to the console. That data often must then be imported into a graphing package to be useful. These tasks have little to do with developing skills in designing engineering models, or in implementing the numerical methods required for the solution, or in understanding the challenges associated with the numerical techniques.

This method, "template based" programming, provides the students with a nearly complete program. The program contains fully debugged input and output routines, including limited graphical analysis in the program output. The student provides the model equations in a form appropriate to the program, and is usually asked to solve the model using numerical methods they implement or by calling appropriate library routines. It is worth noting that this differs from programming by example, since there is no code provided in the specific routines the students are required to write. Furthermore, the method yields an attractive and useful final program that students may choose to further develop and use for solution of problems elsewhere in the engineering curriculum.

II. Background

The 1997 CACHE Corporation Computer Survey of chemical engineering programs indicated that 95.4% of ChE programs teach programming courses[5]. FORTRAN is the most common language taught in these courses[5]. An earlier CACHE Corporation survey provides an interesting companion statistic. Of the practicing chemical engineers surveyed, 92% never use FORTRAN or

another computer language in their work[6]. Furthermore, 86% of employers did not expect literacy in different computer language paradigms.[6] With statistics such as these, it is unclear what specific core computing skills are required for practice in academia and in industry.[7]

If practicing engineers are not programming, why do we teach programming courses? It is because developing a program requires that the program author break down a problem into a logical series of steps in a rigorous language, with a flow of logic that should reflect the mental discipline an engineer must be capable of to solve challenging problems. Programming concepts strengthen two key facets of engineering education, problem formulation and problem solving.[8]

Programming skills are therefore not critical to chemical engineering as a final goal, but instead are a means to the end of developing engineers capable of solving complicated real problems. Stephanopoulos et al. states "a chemical engineering student or practitioner should not need to master all intricacies of FORTRAN 90 or ANSI C in order to create and run computer programs."[8] Indeed, programming languages are a "novel formal medium for expressing ideas about methodology."[9]

If it is accepted that programming, but not the complete application development process, is important, then there must be some components of program development are superfluous to chemical engineering education. Elimination of the superfluous components, while retaining programming elements sufficient to accomplish the task of developing the analytical and communication skills of engineering students, should be a goal of the engineering curriculum.

One approach to addressing this problem is utilizing the scripting capabilities of general-purpose mathematical software, such as MATLAB, Waterloo Maple[10], or Mathematica. This approach has some appeal, since much of the "overhead" of programming is handled by the host application. When the topic of coverage is a fundamental numerical method, however, use of a high level programming language may be a more effective approach. In other cases, implementation of a model and its solution may be as challenging within a mathematics package as with a high-level language. Additionally, in curricula where a high-level language such as FORTRAN is taught, there may be little opportunity to teach programming within another environment.

The use of templates to "trim the fat" from the student programming task is not restricted to high-level languages. FORTRAN instruction is currently required of all University of Kentucky chemical engineering students; therefore it is logical for us to utilize the skills students have already developed in the first implementation of template-based assignments.

The course in which this approach is first being used is entitled Process Modeling. The University of Kentucky course catalog describes this course as "Applications of principles of material and energy balances, thermodynamics, heat and mass transfer, physical chemistry and numerical methods to problems in separation and transport processes and reactive systems."[11] Within the course as offered by the author are several FORTRAN programming assignments requiring coding of numerical methods followed by the solution of student-developed models.

III. Method

The construction of a template requires a determination of the expected outcomes from the activity, so that unnecessary programming components are eliminated. In this course, the student is expected to design a model, including specifications of what is known and unknown for a particular problem. After the model is prepared, the student must implement the model equations as a function or set of functions in a program, and to write code for numerical methods suitable to solve the model. The student is not expected to: write routines to collect model parameters from a console or data file; design structures to pass data between routines; format numerical output and send to console or file; or visualize the data. The student will need to debug his code, which may be simplified by inclusion of debugging routines in the template. The student must validate both the model and the numerical method.

This approach yields a completed project in which the student has focused on the objectives of the course—in this case, modeling chemical processes, then selecting and implementing appropriate methods to solve the models. He has not gained experience in formatting output, creating variable structures, exploring inheritance or polymorphism, choosing between random or sequential data storage, or a host of other worthwhile topics-- in another context.

Note that no specific computer language was mentioned in this description. This approach would be as valid for a UNIX system as it is for a Microsoft Windows[12] or MS-DOS[12] system. It applies to programming in Waterloo Maple or MATLAB or Compaq Visual FORTRAN[13] or Microsoft Visual BASIC[12]. The language is not important, provided the student has adequate background or instruction. The focus on desired outcomes within an engineering course is the benefit to the student.

Additional benefits of using templates include: a standardized product to evaluate for grading purposes; consistent data analysis; and a user-friendly and attractive program. The student also has a complete, functional, and modular application program that the students understands and can use as an example for future application development. For FORTRAN, Visual BASIC, or C++ templates under Windows (or windowing UNIX environments), GUI applications give numerous opportunities to enhance student understanding of numerical methods and models beyond console applications alone. The example of GUI application code may also motivate some students to learn more about programming on their own through independent experimentation.

IV. Implementation

The course in which this approach was first implemented was Process Modeling, a course usually taken in the second semester of the third year. One of the expected outcomes for the course is that students will be able to “use computer software and programming languages to solve complex mathematical systems.” During previous offerings of the course, students were required to write

several FORTRAN programs, including programs implementing routines for solving systems of linear equations and ODE's, and a program investigating limitations of numerical solutions on a computer.

The first time programming templates were used, students were required to write the same programs as had been assigned the previous semester, only this time they were provided with templates for a GUI version of the program. All students chose FORTRAN over other available options, including MATLAB, Maple, Mathsoft MathCAD[14], Visual BASIC, and Microsoft Visual C++[12]. This is likely due to the curriculum requirement of completing a course in FORTRAN programming in order to obtain upper class standing.

The first part of each assignment required the student to derive the model (if applicable) and present a list of all necessary input and output variables to both the model equation function (or functions) and the numerical method. The second part required the student to write both the model function and the numerical method subroutine. The students were provided with all files related to the application, including a code-free subroutine containing comments defining the variables passed to the subroutine and special functions available within the application. After completing the assignment, students submitted an electronic copy of all programming files. Students also submitted a brief report of their results, along with comments, conclusions, and other components of the assignment.

V. Assessment

The class sizes at the University of Kentucky engineering programs at Paducah are small. The current Process Modeling class has four students enrolled. Consequently, statistically validated assessments are impossible.

Common assignments in consecutive offerings allow some qualitative comparison. Students were able to successfully complete their assignments with considerably less assistance from the instructor. The instructor was required to provide extensive assistance to students during previous offerings of the course, primarily in fundamentals of writing input and output, and passing variables between subroutines. Students still required assistance with programming logic, but the time spent working with students on the programming task was reduced.

Surveys were conducted at the start of the course and following the completion each programming project. The initial survey covered general topics such as the student's overall comfort level with writing FORTRAN programs, as well as specific knowledge areas, such as understanding of passing of data arrays to subroutines. Project surveys repeated selected questions related to the project from the initial survey, in addition to questions regarding time usage and satisfaction with the template approach. Summary data for selected questions is provided in Table 1.

Survey Question	Initial Survey	Post Project 1	Post Project 2
I am comfortable writing programs using FORTRAN.	1.75	2	2.5
I understand how to use variable arrays in FORTRAN.	1.25	N/A	2
I understand how to use if-then else structures in programs.	2.75	3.5	3.5
I understand the use of DO loops and how they are terminated.	3.25	3.75	3
I understand the use of comparison operators in FORTRAN.	3.25	3.5	4
I can perform mathematical operations on variables in a FORTRAN program.	3.25	3.75	3.25
Using a template allows me to focus more on the engineering problem compared to writing a program from a blank file.	N/A	4	3.75
The time spent programming was reduced by using the template.	N/A	4.25	4.25
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	4.25	4.5

Table 1. Summary of survey results for FORTRAN template-based projects. A score of 1 indicates student "strongly disagrees" with the statement, and 5 indicates strong agreement. Sample size was 4. All students responded to all questions. Results from third and final survey will be available at a later date.

Two points are gleaned from the survey. First, students are developing their programming skills and increasing their personal comfort level with programming and programming concepts. Second, the use of templates is perceived as a more efficient use of time, allowing a stronger focus on numerical methods and modeling.

Anecdotal evidence is also available. Students were far more willing to attempt the programming task than in previous offerings. They were pleased with the visual results of their efforts, and claim better understanding of the programming that they did accomplish. Students from previous course offerings indicated that use of templates in their courses would have been beneficial.

The most costly aspect of this approach is the time required to write and debug the programming templates. However, once the template is written, it is readily modified for other assignments.

VI. Further Development

The student interest and appreciation of this approach leads to the obvious suggestion to expand the use of programming templates to other chemical engineering courses in which a programming assignment is required. This approach will next be taken in future offerings of the reactor design course and the material & energy balances course.

Exposing students to other programming languages and operating environments may benefit

students as they move into their first job and adapt to whatever computing infrastructure their employer utilizes. With the job market for computer programmers so strong, chemical engineers who are prepared to handle some maintenance and debugging tasks on existing computer systems may be of particular value to industry. Students should have some familiarity with programs such as MATLAB, as well as a high-level language, and some understanding of networking environments, particularly web based environments. All of these environments could be introduced to the students with the use of templates in the context of their chemical engineering courses. The templates will relieve the necessity of the students becoming "experts" with every software package they wish to program, while enabling them some experience with different approaches to programming.

Another course under consideration is an elective that would consist in part of introducing students to a wide variety of computing systems and environments in a chemical engineering context. Students might implement models in FORTRAN, C, or Java[15], develop an interface in MATLAB or Visual BASIC, and then learn how to port all of this to a format allowing web-based access. Writing custom unit operations for Aspen Technology ASPEN Plus[16] or Chemstations ChemCAD[17] could also be required of students using the template approach. Programmable data acquisition software is another potential target platform for the use of templates.

VI. Conclusions

Template-based programming reduces the amount of effort required of students toward outcomes not tied to the objectives of a chemical engineering course. The reduction in time required for the programming task increases the available time to focus on understanding model development and appropriate usage of numerical methods. The approach is not limited to a particular language or environment, and may be utilized in a variety of courses. The biggest drawback to the technique is the substantial faculty time that may be required to prepare a robust template for student use.

Bibliography

1. URL: http://www.mathworks.com/; The Mathworks, Inc.
2. URL: http://www.wolfram.com/; Wolfram Research, Inc.
3. Smart, J.L., Murphy, W., Lineberry, G.T., & Lykins, B. Development of an Extended Campus Chemical Engineering Program. Proceedings of the *2000 ASEE Annual Conference & Exposition.* American Society for Engineering Education, (2000).
4. Capece, V.R., Murphy, W., Lineberry, G.T., & Lykins, B. Development of an Extended Campus Mechanical Engineering Program. Proceedings of the *2000 ASEE Annual Conference & Exposition.* American Society for Engineering Education, (2000).
5. URL: http://www.che.utexas.edu/cache/survey.html; CACHE: Survey Results
6. Davis, J., Blau, G., & Reklatis, G.V. Computers in undergraduate chemical engineering education: A perspective on training and applications. Technical report, CACHE Corporation. Draft 3.1. (1993).
7. Kantor, T.J., Edgar, T.F. Computing skills in the chemical engineering curriculum. In B. Carnahan (Ed.), *Computers in Chemical Engineering Education*, Austin, Texas: CACHE Corp. (1996).

8. Stephanopoulos, G. & Han, C. Languages and Programming Paradigms. In B. Carnahan (Ed.), *Computers in Chemical Engineering Education*, Austin, Texas: CACHE Corp. (1996).
9. Abelson, H. & Sussman, G. Structure and Interpretation of Computer Programs, Cambridge, MA, MIT press (1985).
10. URL: http://www.maplesoft.com/; Waterloo Maple, Inc.
11. University of Kentucky 2000-2001 Bulletin, Lexington, Kentucky: University of Kentucky (2000).
12. URL: http://www.microsoft.com/; Microsoft Corporation.
13. URL: http://www.compaq.com/fortran/; Compaq FORTRAN Products.
14. URL: http://www.mathsoft.com/; Mathsoft, Inc.
15. URL: http://www.sun.com/; Sun Microsystems, Inc.
16. URL: http://www.aspentech.com/; Aspen Technologies.
17. URL: http://www.chemstations.com/; Chemstations.

DAVID L. SILVERSTEIN

David L. Silverstein is currently an Assistant Professor of Chemical and Materials Engineering at the University of Kentucky College of Engineering Extended Campus Programs in Paducah. He received his B.S.Ch.E. from the University of Alabama in Tuscaloosa, Alabama, and his M.S. and Ph.D. in Chemical Engineering from Vanderbilt University in Nashville, Tennessee. He has over twenty years experience in microcomputer programming, most recently in development of a prototype automatic custom videotape editing and production device. In addition to teaching, Dr. Silverstein is developing a computer framework for applying teaching styles to a multimedia computer based supplement to engineering courses.