USING A READILY AVAILABLE COMMERCIAL SPREADSHEET

To Teach a Graduate Course on Chemical Process Simulation

MATTHEW A CLARKE AND CARLOS GIRALDO
University of Calgary • Calgary, Alberta, Canada T2N 1N4

For chemical engineering graduates, being able to use a chemical process simulator is considered as a *sine qua non* of the discipline, yet relatively few students have a direct appreciation of what is involved in constructing a chemical process simulator. The complex chemical process simulators, such as ASPEN and HYSYS, that are almost universally known to chemical engineers, have significantly streamlined the task of chemical process design. Efficient these simulators are, however, they also mask myriad complex calculations. Thus one of the end results is that users of these software packages may not have a full appreciation of how elegantly and succinctly process simulators intertwine almost all aspects of chemical engineering from thermodynamics to equipment design to cost estimation.

An ancient Chinese proverb says, "Tell me and I'll forget; show me and I may remember; involve me and I'll understand," and it was with this mindset that Professor P.R. Bishnoi (one of the founders of Hyprotech) developed, in the mid-1980s, a post-graduate course in process simulation in which students would enhance their understanding of process simulation by constructing all, or part, of a chemical process simulator. In the initial years of its existence, individual students constructed relatively simple simulators, using FORTRAN 77, to solve a specific problem rather than to serve as a general process simulator. In the 1990s, there was a shift from FORTRAN 77, a procedure-oriented language, to C and then finally to C++, which is an object-oriented language. The use of an object-oriented language allowed for the creation of re-usable blocks of code that could be connected in a near infinite number of configurations, thereby greatly extending the generality of the students' simulator. As the scope of the term project became more complex over the years, it necessitated a migration from individual term projects to a format in which all of the class member would contribute to a single final project. To the best of the author's knowledge, this course is unique and is not offered, in this format, at any other institution.

Following Professor Bishnoi's retirement, this course took an extended hiatus from the list of courses available to graduate students in Chemical and Petroleum Engineering at

Matthew Clarke *is an assistant professor of Chemical and Petroleum Engineering at the University of Calgary. He has a Ph.D. in chemical engineering from the University of Calgary. His research interests are in hydrates and biofuels.*

Carlos Giraldo *is pursuing his Ph.D. in chemical engineering at the University of Calgary. He graduated from the National University of Colombia in 2001 with a degree in chemical engineering. Subsequently, he worked as a process engineer in Bogota, Colombia.*

the University of Calgary. At the time of its resurrection, in 2007, much had changed in the world of chemical engineering education and computing, particularly with respect to students' familiarity with structured programming languages. Thus, the decision was made to do the project with a more familiar platform: that of MS Excel with VBA. While it is not possible to construct a stand-alone simulator using Microsoft Excel, it was an attractive choice because it provides a convenient platform for entering and displaying data and because it is almost universally known to post-graduate students in chemical engineering. The use of spreadsheets for various chemical engineering calculations is not new and has been documented in many sources.[1-5] What makes this course unique is that Excel is being used to create an "all-purpose" simulator, with a built-in graphical user interface (GUI), for solving simultaneous heat and energy balances.

Additionally, unlike traditional graduate courses in chemical engineering, this course is distinctive in that it is a group effort, rather than a solo pursuit. Because of the sheer size of the project, it is not feasible to expect each student to construct his or her own simulator in a one-semester course. Thus, each student is given a task and, toward the end of the semester, the pieces are fit together to produce a working simulator.

Because of the unique nature of this course, this paper is being written to evaluate the delivery format of the course and the effectiveness of teaching chemical process simulation in a "hands-on" format.

COURSE DESCRIPTION AND DELIVERY FORMAT

The idea of learning by doing is not a new one; the great Italian renaissance polymath Leonardo Da Vinci once said "Knowing is not enough; we must apply. Being willing is not enough. We must do." The objective of this course is to give to the students hands-on experience at developing a general-purpose, steady-state, chemical process simulator and to understand the various types of computations involved in such a simulator. In this respect, it's as much a course in software design as it is a course in chemical engineering.

In the university's calendar, the course was allocated three hours per week for lectures, of which half was used for lecturing and presenting new material and the other half was used for a weekly seminar in which students would individually discuss their previous week's progress and their goals for the following week. This seminar was particularly important since two of the students were employed full time. The material covered in the lectures was, for the most part, not entirely new to the students and consisted of a review of basic heat and energy balances and unit operations, selected topics from numerical methods and computational thermodynamics, and sequencing. Sequencing, which will be discussed in more detail later, was the only section that was completely new to the students. At several points throughout the semester, an invited speaker who worked in the chemical process simulation field gave lectures on object-oriented programming fundamentals and on how to put all of the components together to form the final product.

Because the course was primarily a group effort, evaluation of the students' progress was not a trivial task. As alluded to earlier, each student was given a specific task to work on and then at the end of the semester, the pieces were assembled into their final magnum opus. Thus, for roughly the first two-thirds of the semester, each student worked independent of the others and it was felt that the appropriate method for mid-term evaluation was a combination of a written report and an oral exam for each student. Theses components were each worth 15% of the final grade. In the latter portion of the semester, the students were collaborating much more closely than in the early part of the semester and thus the term-end evaluation consisted of a final oral presentation and exam for each student, worth 25% of their grade, plus a single final report submitted by all group members, worth 45% of the final grade.

WHAT THE STUDENTS THOUGHT THEY WERE TAKING

The aforementioned course was first offered to graduate students more than 20 years ago. Due to the original instructor's retirement, however, the course had not been offered in almost five years. Thus, students were not able to learn about the course by word of mouth and the only information available to the students regarding the course came from the university's catalog description, which, at the time read

> *"Chemical Process Simulation: Synthesis. Analysis and screening of process alternatives. Steady state simulation. Material and energy balances for systems of process units. Modular approach. Heat exchanger network and separation processes."*

When this course description was first conceived, two decades ago, it quite adequately conveyed what the course was about. Most of the 12 students that arrived to class on the first day of the semester were expecting something completely different, however. When I asked the students what they were hoping to achieve in taking this course, responses ranged from "HYSYS training" to "I need an easy course for my professional registration." When I handed out the course outline to the students, a great silence fell across the room and faces paled at the revelation that this was not a software training course, that this would not be an easy course, and that they would have to do the programming themselves. The end result was that the class size dropped from 12 to four by the second lecture. (It should be noted that, because of the necessary division of labor, four students is the minimum number necessary to run the course.)

After re-examining the calendar description quoted above, I felt it could be written to better elucidate the course content. For the upcoming year, the calendar entry has been changed to

make it explicitly clear that it is an object-oriented programming course. It now reads:

> *"Chemical Process Simulation: Object-oriented programming applied to the design of a steady state chemical process simulator via the sequential modular approach and by the equation-based approach. Material and energy balances for systems of process units."*

It was felt the above wording would not only inform students that it is not a software training course, but it would also inform other students that it is a programming course.

WHAT SKILLS THE STUDENTS HAD TO LEARN

The students enrolled in the course had a diverse academic background and each brought certain strengths to the project. As previously mentioned the lecture content included topics from basic chemical engineering, numerical methods, thermodynamics, and sequencing. Thankfully, all students were well versed in the basics of chemical engineering and numerical methods and thus it was not necessary to devote significant lecture time to these topics.

Computational thermodynamics, on the other hand, was a subject with which the students had little direct experience. Thus, this section included a review of vapor/liquid equilibrium, equations of state, activity coefficient models, bubble point and dew point calculations for nonideal systems, isothermal/isobaric flash calculations, isoenthalpic/isobaric flash calculations, isentropic/isobaric flash calculations, and Gibbs free energy minimisation for reacting systems.

Sequencing, as mentioned previously, was the lone topic that was completely new to the students and the lectures were devoted to presenting various algorithms for determining in which order the unit operations should be solved.

The biggest deficiency in the students' knowledge base, however, was not in any of the chemical engineering topics; it was in object-oriented programming. Maixner[6] observed that after taking basic engineering computing, students "usually allow their programming skills to stagnate." While some of the students were out of practice with respect to programming, it was found that for 100% of the students, their knowledge was limited to procedure-oriented programming. The advantage of an object-oriented approach is the ability to solve and test individual modules together and the ease with which modules can be combined, solved, analyzed, and swapped.[7]

After the semester, the three remaining students were asked, among other things, what skills they had to learn and their responses are listed below:

- *"Thermodynamics, unit operations, programming, logical thinking, teamwork."*
- *"During the course, we learned some concepts of OOP, thermodynamics, simulation solver other than extensive use of computer programming."*
- *"Teamwork."*

ORGANIZATION OF THE SIMULATOR

The construction of a steady state, sequential, modular simulator is a daunting task when first approached by students and it's not feasible for individual students to construct their own simulator in a one-semester course. Thus, the project was done as a group project in which each student was assigned a part of the object-oriented simulator to construct. Initially, one student was assigned to create a GUI and component database, one student was assigned to create the thermodynamic routines, one person was responsible for the unit operations, and the remaining member was responsible for sequencing. Midway through the semester, the student in charge of the thermodynamics routines withdrew from the class and these duties were subsequently divided among the remaining group members. Fortunately, one of the remaining students had previous exposure to computational thermodynamics.

The program is composed of a GUI, information sheets, thermodynamic routines, unit operations routines, and the sorting/tearing algorithm—all of which interact with each other, as shown in Figure 1. In the ensuing subsections, a brief description of each section of the program is provided.

Information Sheets

The information sheets, which are merely Excel worksheets, provide a platform for entering and displaying data and for storing thermodynamic component data. Data is read from and written to the appropriate worksheet using the built-in *cells()* function.

GUI and Component Database

The graphical user interface was constructed in Excel using the built-in VBA editor. Once the equation of state and the components for the new project have been selected, the program closes all dialogs and takes the user to the main interface window, where the Process Flow Diagram is built.

On this page, the program shows the core of the graphical user interface: the "Unit Operations" tool bar. This tool bar is divided in five main sections: adding new unit operations,

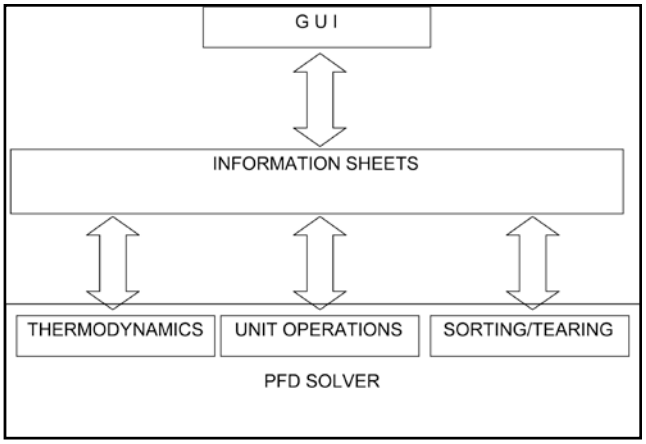


Figure 1. *Block diagram for the simulator.*

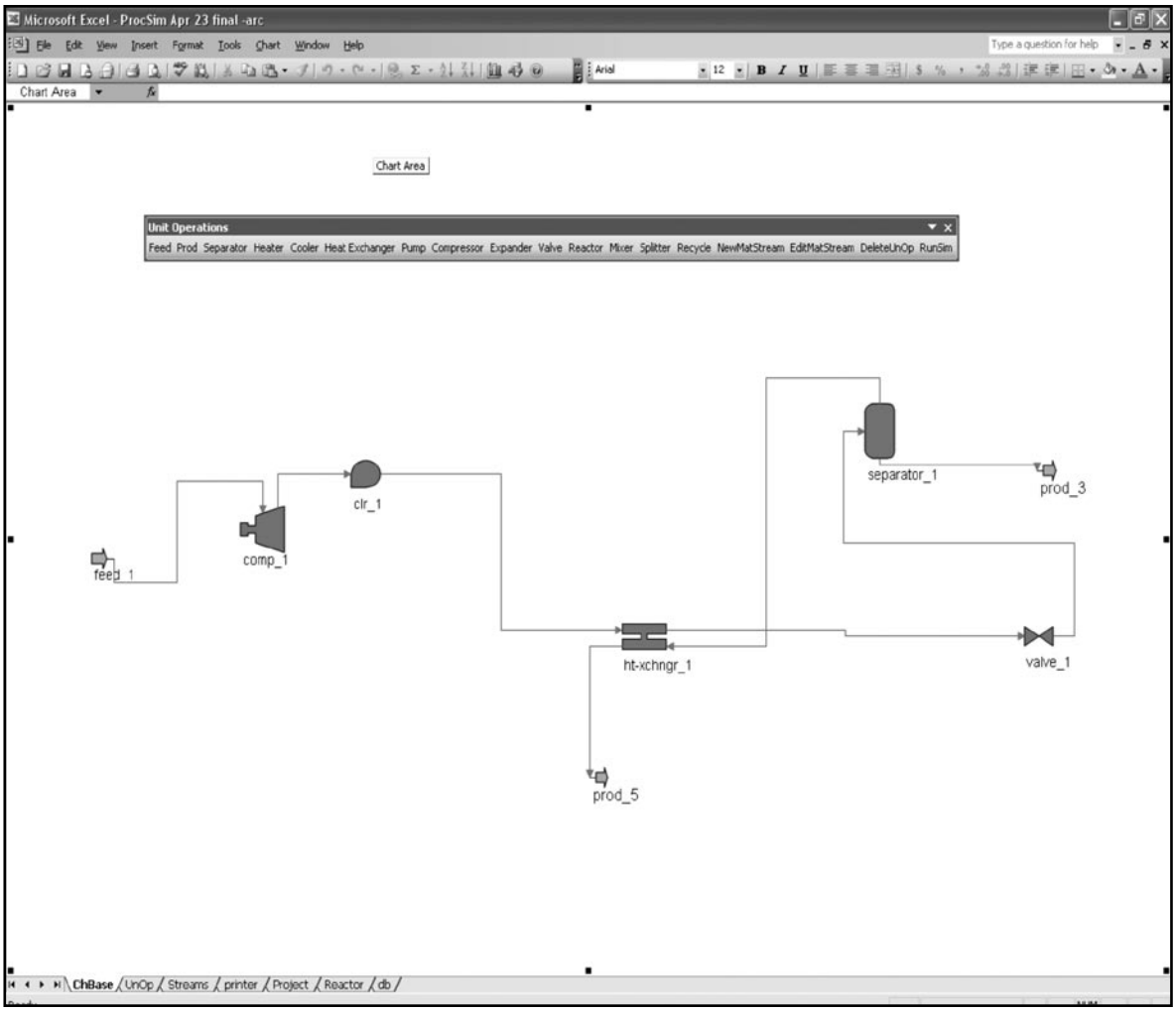
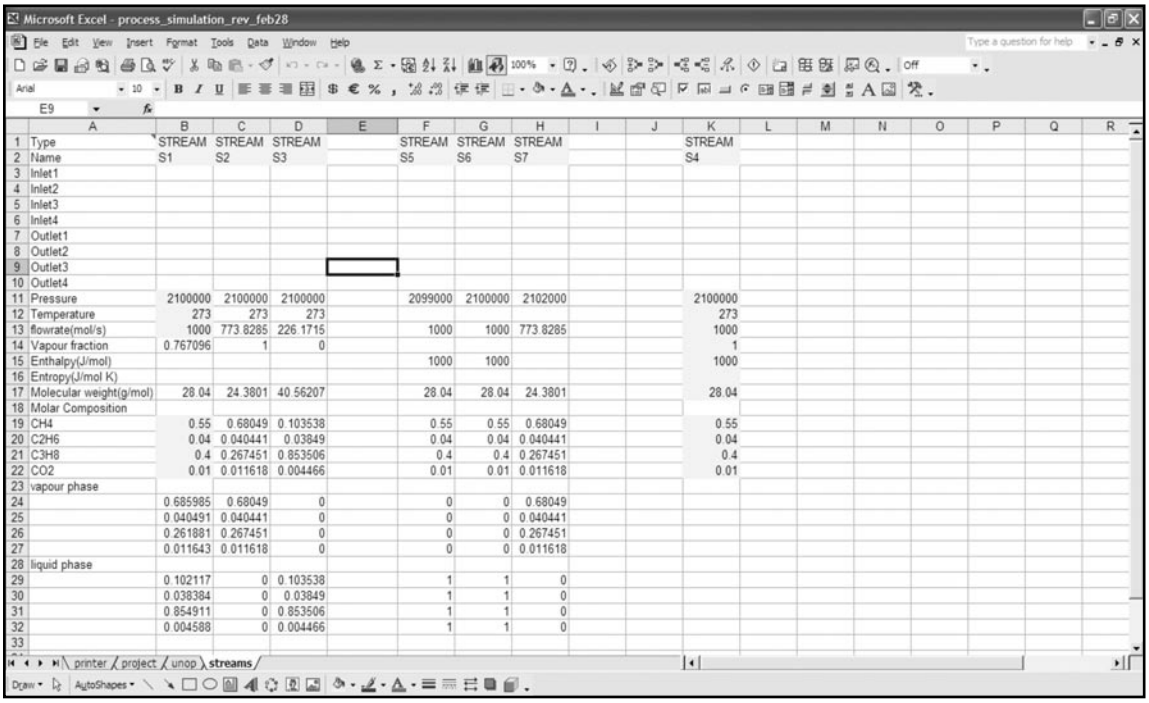


Figure 2. *Screen capture of a typical flow diagram.*



	A	B	C	D	E	F	G	H	I	J	K
1	Type	STREAM	STREAM	STREAM		STREAM	STREAM	STREAM			STREAM
2	Name	S1	S2	S3		S5	S6	S7			S4
3	Inlet1										
4	Inlet2										
5	Inlet3										
6	Inlet4										
7	Outlet1										
8	Outlet2										
9	Outlet3										
10	Outlet4										
11	Pressure	2100000	2100000	2100000		2099000	2100000	2102000			2100000
12	Temperature	273	273	273							273
13	flowrate(mol/s)	1000	773.8285	226.1715		1000	1000	773.8285			1000
14	Vapour fraction	0.767096	1	0							1
15	Enthalpy(J/mol)					1000	1000				1000
16	Entropy(J/mol K)										
17	Molecular weight(g/mol)	28.04	24.3801	40.56207		28.04	28.04	24.3801			28.04
18	Molar Composition										
19	CH4	0.55	0.68049	0.103538		0.55	0.55	0.68049			0.55
20	C2H6	0.04	0.040441	0.03849		0.04	0.04	0.040441			0.04
21	C3H8	0.4	0.267451	0.853506		0.4	0.4	0.267451			0.4
22	CO2	0.01	0.011618	0.004466		0.01	0.01	0.011618			0.01
23	vapour phase										
24		0.685985	0.68049	0		0	0	0.68049			
25		0.040491	0.040441	0		0	0	0.040441			
26		0.261881	0.267451	0		0	0	0.267451			
27		0.011643	0.011618	0		0	0	0.011618			
28	liquid phase										
29		0.102117	0	0.103538		1	1	0			
30		0.038384	0	0.03849		1	1	0			
31		0.854911	0	0.853506		1	1	0			
32		0.004588	0	0.004466		1	1	0			
33											

Figure 3. *Screen capture of the "Streams" worksheet.*

adding a material stream, editing a material stream, deleting a unit operation, and running the simulation.

Thermodynamics

Thermodynamic calculations represent the basis of the simulator, since all the unit operations are assumed to reach the equilibrium. To develop the current simulator, two equations of state have been selected: the Soave-Redlich-Kwong (SRK) equation of state[8] and the Peng Robinson (PR) equation of state.[9] As part of the process simulator, we are generating the value of thermodynamic properties (enthalpy and entropy), thus, the designation of a reference state becomes essential. For this simulator, the reference state that was chosen was an ideal gas at 273.15 K and 1 Pa.

The thermodynamic routines provide a necessary support for the unit operations in that they calculate necessary outlet conditions such as temperature, pressure, and/or composition. Depending upon the specific unit operation, these values are calculated with the aid of a constant-temperature/constant-pressure flash routine, a constant-enthalpy/constant-pressure flash or a constant-entropy/constant-pressure flash. The values of the thermodynamics quantities, such as entropy and enthalpy, are calculated by using the appropriate departure function[10] in conjunction with an equation of state. In addition, a Gibbs free energy minimisation routine was implemented to support the reactor unit operation. The algorithms for each of these routines are available in the open literature.[10-12]

Unit Operations

To code the unit operations, an object-oriented programming philosophy was adopted. For this section it is important to introduce some related terminology, mainly the object, class, properties, and method. An object is defined as a programmable entity with specific characteristics and feautures,[13] a class is what defines the object and serves as the template or blueprint for all the objects created from that class; any of the object's particular features, including properties, methods, and events, are handled by the class module.[13] Properties are all the characteristics and features of the object without executing any action, and method is an entity that provides the actions supported by the objects created from the class.[13] Methods are defined as the abilities, or actions, that the object can carry out.[13]

Basically, a module class was created for every single unit operation; this class contains the most important properties and methods. In addition, modules were included to collect the inputs from the interface, which become the properties of the class and present the results again on the interface. Unit operations that were included in the project were separators, valves, pumps, compressors, expanders, heaters, coolers, heat exchangers, mixers, splitters, recycle loops, and a Gibbs free energy minimization reactor. The details of the individual unit operations are available in any standard chemical engineering textbook.[14]

Sequencing

Sequencing is a term that essentially refers to the process of figuring out the order in which each of the unit operations should be solved. The sequential modular approach is the most popular approach and there are a number of commercial simulators that use this solution methodology. "Sequential modular" simply means that calculations start with known feeds, and continue on a unit-by-unit basis until all unit operations in the flowsheet are calculated.[15]

In sequential modular approach, the material balances for an entire process are solved one module (process block) at a time. For sequential modular material balance calculations, the output streams can be calculated if the input streams and the module parameters are known.

VALIDATION OF THE NEW SIMULATOR

A case study was initiated to study how the new Process Simulator compares with the popular commercial simulator, Aspen HYSYS. The simple case illustrated the capability of the new Process Simulator for performing reliable chemical process calculations for the Linde Process. Methane is usually liquefied in a Linde process. Initially, the vapor is compressed to 6MPa and cooled at 300K. Subsenquently the vapor passes through a heat exchanger before reducing its pressure (to 0.1MPa) through a valve. The unliquefied fraction leaves the separator at the saturation point and passes through the heat exchanger exiting at the end at 295K. A screen capture of the flow diagram and the constructed worksheet for the process used in the case study is presented in Figures 2 and 3.

The components that were present in the case study were methane, ethane, propane, n-butane, and carbon dioxide. T he fluid properties were calculated with the PR equation of state. It was found that all of the stream compositions, temperatures, and pressures as calculated by the new simulator were within 1.5% of the values calculated by the commercial simulator.

EVALUATION OF THE COURSE FROM A STUDENT'S POINT OF VIEW

As previously mentioned, at the conclusion of the course, the students were asked for their feedback. In addition, one student (who is the second author on this paper) provided a detailed evaluation of the course. The following paragraphs are his evaluation of the course:

> *"The construction of a steady state, sequential modular simulator represented an immense endeavour challenge for all the members of the group since none of us had done anything similar in previous courses. The experience acquired until that moment with programming languages was limited to very "simple" algorithms dedicated for particular cases. Thus, when the distribution of the work was assigned by the professor, a sensation of confusion came up in the group because nobody was really sure about how to tackle the problem. Since the beginning of the semester, this course*

was different from the ones a regular graduate student usually takes, where the professors impart lots of information in the lectures. In this case, the classes were divided in two sessions per week; one of them was a lecture reviewing the basic tools that we required to complete this course successfully and the second one was a group meeting to show everyone's progress, difficulties, and critical points, etc.

During the first few weeks, the work was done independently; each of the members of the group wrote his or her own code without considering anybody else's problems or difficulties. Integration of the code was not the main concern in that particular moment. But, fortunately, the professor invited as guest lecturer a former student who had taken the course few years ago; he basically commented about his experience in creating a process simulator. His main contribution to us was definitely the concept of working as a group and the fact that it is necessary to define completely the structure of the simulator on a piece of paper before any coding is done; in addition, he highlighted that the final product should be written thinking that it is going to be used by an external user. From that moment on, teamwork began and it was established how the simulator was going to work and what kind of special characteristics should be included. For instance, definition of the variables that would be supplied by the user, the variables that would be public (available for the all program) or private and the possible inconsistencies due to errors when wrong data were supplied by the users, final outputs, etc.

Once, the designing stage was finally done, all of the members of the group had to recode a big part of our their programs to make them suitable for the final integration. After having succeeded on this, the skeleton of the whole simulator was ready. The following phase was to complete the rest of the thermodynamic algorithms and unit operations, which was a lot easier due to main parts being already done. The first simulator was really big and it was really difficult to handle, thus, some parts of the program to were recoded once again some parts of the program to make them better. In this part of the work a very helpful support was provided from one student at U of C whose knowledge in Visual Basic for Applications was far beyond any of the other members of the group. His main contributions were the implementation of useful techniques and hints to reduce the number of lines in the program and increase the speed of processing. One of the most challenging parts was the debugging of the code; from our point of view the best way to debug any long program is by ensuring that every piece of code is working properly rather than to verify the whole program. It is also important to keep in mind that a minimum knowledge is required of programming, creation of algorithms, numerical methods focused on the solution of single and simultaneous systems of non-linear algebraic equations, sorting, matrixes, etc. But the most important thing is to always be motivated and be open to learn new concepts. At end of the term, the group was really satisfied with the final product; since we were able to reproduce the results of commercial simulators by using simple programs."

Additional comments by the other students are given below:

Which parts were the most challenging?

- *"Finding the best way to represent the unit operations and connect the graphical part to the database in excel. "*
- *"The most challenging part was debugging. The algorithm initially written was not the best one. The program went very big and was just too big to handle."*

Problems to comment on for future students?

- *"Teamwork is a key component in this course."*

Which skills you got to the end of the course?

- *"Computer programming skills, basic structure of simulator, simulation solver knowledge. Above all we learned how to put together a simulator."*

EVALUATION AND RECOMMENDATIONS FROM THE INSTRUCTOR'S POINT OF VIEW

The structure and content of the course make it unique among graduate courses in chemical engineering at the University of Calgary and thus it presented many challenges to the instructor. As previously mentioned, this was the first time in almost five years that the course had been offered, and in spite of that fact, the overall sentiment of the instructor was that the course achieved its desired outcomes.

Over the course of the semester, there were only two negative events to dampen evaluation of the course. The first was that, as previously stated, a large number of students enrolled believing the course would be a software-training course. It is also hypothesized that some students—who may have been keen programmers—avoided the class for the same reason. Secondly, the number of students was below the optimal number for the course. When one of the students withdrew from the course, a tremendous burden was placed upon the remaining students.

Conversely, many aspects of the course were extremely positive. The total project was divided into manageable parts and, at the end of the semester, each student was able to say that they had not only learned how to program a part of the simulator, but they also benefitted from having to work in a team environment. The regular weekly debriefings not only allowed the students to receive weekly feedback from the instructor, but they also served to educate each group member as to what challenges the other members were facing with their programming. Additionally, the use of Excel as the programming platform was advantageous. While Excel may not have all of the capabilities of high-level programming language, such as the ability to create a stand-alone executable file, the fact that students had previous exposure to it and the fact that student access to Excel is practically universal made it the best choice for the course.

For any instructor that is contemplating offering a graduate course in simulation, I would make the following recommendations:

1. *Ensure that the calendar description clearly conveys that it is a course in software design and not merely a software-training course.*
2. *Advertise a detailed synopsis of the course well in advance of the beginning of the term.*
3. *Stipulate a minimum of six students. This ensures that if one or two students withdraw partway through the term, the remaining students are not overly burdened.*
4. *To attract and retain a larger number of students, include minimal hands-on work with a commercial simulator, perhaps as a means of illustrating lecture topics.*
5. *Spend the majority of lecture time covering object-oriented programming fundamentals, computational thermodynamics, and sequencing.*

CONCLUSION

A post-graduate course on chemical process simulation was offered in the Department of Chemical and Petroleum Engineering at the University of Calgary in which students were given the opportunity to construct a chemical process simulator as a group project. Microsoft Excel, along with its built-in Visual Basic for Applications (VBA) programming environment, was used to create a fully functioning modular chemical process simulator. An object-oriented approach was used to create and combine the necessary sections of the simulator; mainly the graphical user interface and component database, the thermodynamic routines, the unit operations, and the sequencing algorithms. The result is a fully functioning, steady state, chemical process simulator that is capable of matching the results of the far more expensive commercial simulator, as was seen with the validation study. Student feedback on the course indicated that the students learned a great deal with respect to both software design and to working in a team environment. From the instructor's point of view, the course was successful in achieving its goals and recommendations for future offerings of the course were made.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. Ryan Krenz from the Virtual Materials Group in Calgary, Alberta, for sharing his experiences in creating object oriented process simulation tools.

REFERENCES

1. Clarke, M.A., and P.R. Bishnoi, "Development of an implicit least squares optimization scheme for the determination of Kihara potential parameters using gas hydrate equilibrium data," *Fluid Phase Equilibria*, **211**, 51 (2003)
2. Lwin, Y., "Chemical equilibrium by Gibbs energy minimization on spreadsheets," *International J. of Engr. Educ.*, **16**, 335 (2000)
3. Savage, P.E., "Spreadsheets for Thermodynamics Instruction," *Chem. Engr. Educ.*, **29**(4), 262 (1995)
4. Ravella, A., "Use a spreadsheet for preliminary reactor design," *Chem. Engr. Progress*, **89**, 68 (1993)
5. Bornt, B., "Spreadsheets for heat loss rates and temperatures," *Chem. Engr.*, **102**, 107 (1995)
6. Maixner, M.R., "Design of a waterjet-propelled barge: a first computer modeling project," *International J. of Engr. Educ.*, **21**, 745 (2005)
7. Chen, J., and R.A. Adomaitis, "An object-oriented framework for modular chemical process simulation with semiconductor processing applications," *Comp. Chem. Engr.* **30**, 1354 (2006)
8. Soave, G., "Equilibrium Constants from a Modified Redlich-Kwong Equation of State," *Chem. Engr. Sci.*, **27**, 1197 (1972)
9. Peng, D., and D. Robinson, "A New Two-Constant Equation of State," *Ind. Engr. Chem. Fund.*, **15**, 59 (1976)
10. Elliot, R., and C. Lira, *Introductory Chemical Engineering Thermodynamics*, Prentice Hall, Upper Saddle River, NJ (1999)
11. Walas, S., *Phase Equilibria in Chemical Engineering*, Butterworth-Heinemann, Newton, MA (1985)
12. Tester, J.W., and M. Modell, *Thermodynamics and its Applications*, 3rd Ed., Prentice Hall, Upper Saddle River, NJ (1997)
13. Harrison, B.K., "Computational Inefficiencies in Sequential Modular Flowsheeting," *Comp. Chem. Engr.*, **7**, 637 (1992)
14. Biegler, L.T., I.E. Grossman, and A.W. Westerberg, *Systematic Methods of Chemical Process Design*, Prentice Hall, Upper Saddle River, NJ (1997)
15. Norman, R.L., "A Matrix Method for Location of Cycles of a Directed Graph," *AIChE J.*, **8**, 450 (1965) ❐