Introduction to Educational Use of Environmental Engineering Software

Aaron A. Jennings
Department of Civil Engineering
Case Western Reserve University

Abstract

Between May, 1995 and August 1997 collaborators from seven Gateway consortium universities worked to develop shared resource modules to help support Environmental Engineering education. This paper discusses how module development priorities were established based on initial prototype results. The paper also discusses the general criteria that were used to identify proprietary professional environmental software products that could serve well in engineering education applications. Details of modules developedfor the selected "impact" areas will be discussed in a series offollowing manuscripts.

Introduction

The work presented here and in the series of papers to follow grew out of an NSF sponsored project to share educational resources among members of the Gateway consortium of universities. The "Environmental Group" of the coalition was formed in the summer of 1995 at a workshop held at Ohio State University, May 22-23. At this workshop, participants agreed to work in three focus groups on a series of projects to explore "shared resource" opportunities built around "Case Studies", "Databases" and "Environmental Software". The concept evolved from the observation that, as individual professors, we were all developing instructional materials in one or more of these areas, but that individual efforts were limited by available resources. The long-term usefulness of some efforts was also believed to be constrained by the difficulties of repeated use. This led to the concept of "shared resource modules". The idea was that we could support one another by sharing resources, and in doing so, expand the value of our individual efforts.

The focus on "Case Studies" was selected because of the observation that it can be difficult to acquire or develop a realistic problem for student design projects. Simplistic problems are easy to find, but do not provide a very realistic vision of the practice of Environmental Engineering. Complete case studies present problems in rich detail. Students can (and must) pour over this information and synthesize a considerable amount of detail before experimenting with potential solutions. Clearly, one must apply this type of assignment with care, but it can be an extremely valuable learning experience as a complement to traditional homework assignments. Unfortunately, it is difficult for one person to repeatedly generate the volume of resource material required for a real case study. Further, problems lose impact with reuse because students begin recycling ideas. Therefore, the idea emerged that if we could share such resources among a larger group of professors, we could all contribute to and benefit from a larger volume of available

resources. Similar arguments were made for sharing expertise about "Databases" and "Environmental Software Packages".

During the summer of 1995, project participants worked in three focus groups to prototype instructional modules build around case studies, databases and environmental software. Each group experimented with module designs, delivery vehicles and testing procedures and investigated the possible barriers (cost, copyright issues, etc.) that might hinder development or distribution. For the purposes of this work, a "course module" was loosely defined as a package of information that would occupy approximately one to two weeks of traditional course time. This was not intended to imply that modules would always be used in traditional courses. We specifically sought materials that could be applied in innovative approaches to education. Rather, the intention was to develop resources in relatively small modular packages that could be used to enrich existing environmental courses, combined to build new courses, or used in innovative educational approaches such as freshman team design projects. It was believed that this approach (as opposed to developing whole new courses) could be rapidly integrated into Environmental Engineering Education across the diversity of ways in which programs are expressed.

Results of Module Prototyping

The products of the summer 1995 effort were presented and evaluated at a workshop held in Newark, N.J., Aug. 24 -25, 1995. Module prototypes were presented, discussed and evaluated. The essential conclusions of this meeting were as follows:

1. In evaluating materials for our three focus group (Case Studies, Databases, and Software) we come to the conclusion that all of these areas are strongly linked and should not be treated as independent entities. Preparing a useful module based on an environmental software code requires that one provide practical examples illustrating how the code can be used. This is best done with a realistic example problem that is, in essence, a case study. Furthermore, analysis codes generally require additional support information. Often this can be retrieved from a data base. Therefore, creative educational use of an environmental software package requires the support of case studies and databases. The same is true for case study or database modules. When students are assigned a case study problem, they are expected to explore creative solutions. Often this requires gathering additional information (e.g. database application) and conducting engineering analysis (i.e. software use). The interactions among these areas are illustrated in Figure 1.

2. There was reasonable agreement that "Databases" did not require the intensity of development that software or case studies required. Databases are important, but once created they generally do not require additional development to be used effectively.

3. It became clear that there was a wide variety of opportunities for module development. There was concern that if our efforts were not focused, we would not produce products that would have the impact we hoped for. Therefore, it was decided that we would focus on a relatively small set of curriculum "impact areas", and concentrate efforts on developing a rich variety of modules in those areas. The following "impact areas" were selected.

I. Solid and Hazardous Waste
II. Water/Wastewater Treatment & Environmental Chemistry
III. Air Pollution
IV. Environmental Hydraulics & Water Resources

These areas were selected because they range from the classical Environmental Engineering topics of Water/Wastewater Treatment to new, rapidly evolving areas such as Hazardous Waste and Environmental Hydraulics. Air Pollution was selected because this is an important subject that is often underrepresented in environmental engineering programs. These areas were also selected because at least one module for each was prototyped during the Summer of 1995.

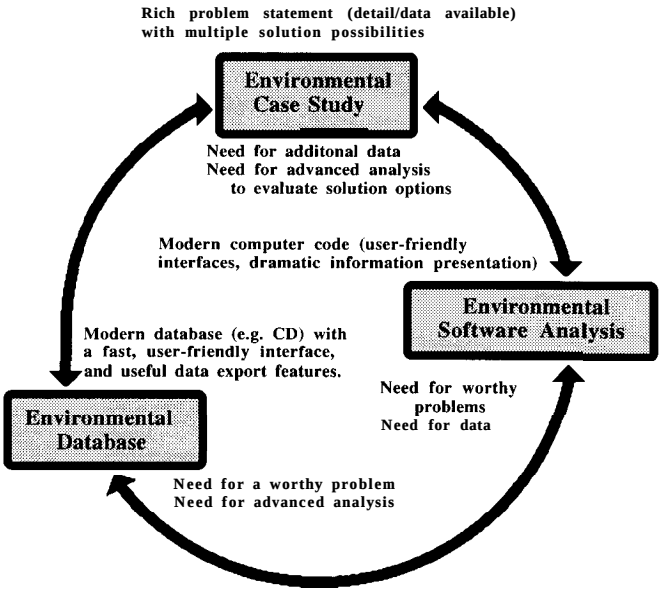

Figure 1 - Synthesis of Case Studies, Databases, and Environmental Software

Results of Module Development

During the following two academic years (Sept. 1995 - Aug. 1997) modules were developed at seven of the ten Gateway Consortium universities. Modules were developed as individual efforts, as multidisciplinary on-campus efforts and by groups of colleagues working at multiple Gateway campuses. The focus of academic year 95/96 was the "alpha" (i.e. on campus) development of a wide variety of modules. In academic year 96/97 the group concentrated on "beta" (i.e. out of developer hands) testing of modules at other participating institutions, and polishing modules into finished products.

The result of this effort was approximately 24 modules. The count is approximate because creativity in model generation produces uncertainty in the rules of quantification (we don't know exactly how to count our results) and because some modules have significant applications in more than one impact area.

Solid and Hazardous Waste . . . . . 10 modules
Water/Wastewater Treatment & Env. Chemistry. . . 10 modules
Air Pollution . . . . . 3 modules
Environmental Hydraulics & Water Resources . . . 5 modules

Modules from each of these impact areas will be the subject of the papers and presentations that follow this introduction (Fentiman and Jennings, 1998; Flora and McAnally, 1998; Roy et al., 1998; Cataldo, 1998).

Software Attributes for Engineering Education Applications

Developing Environmental Engineering instructional modules based on existing, (often proprietary software) can be challenging. One might assume that any software product that is an asset in professional practice would also be helpful for educating students about the problem it addresses. Unfortunately, this is not correct. Software that serves well in professional applications can yield poor results in education. There are several reasons for this, but the most important involve the fundamental nature of the service the software is intended to supply, and the amount of effort (the "learning curve") the user must make before attaining satisfactory results. Those of us who have applied software packages in educational applications have probably all discovered that on occasion, something that seemed like a "neat" idea resulted in a disaster. Although most of us survive these episodes, ultimately they may do more harm than good to our students if they reinforce the wrong messages about computer analysis. It is very important that students do not become frustrated or overwhelmed by software applications. It is also very important that they not become disconnected from the responsibility of making software work properly. People fear a generation of "software user" engineers who don't really know how their design tools work. As educators it is our responsibility to ensure that this does not happen.

The following are desirable attributes for software used in educational applications. They may not all be appropriate for each application, and the list is undoubtedly incomplete, but we have found them to be helpful for evaluating the educational potential of software products. These have been discussed in detail in Jennings (1997) and in a series of papers intended to illustrate how they can improve the educational value of software (Jennings and Kuhlman, 1998; Mesania and Jennings, 1998; Jennings, 1998)

Narrow Focus - It is helpful if the software product focuses on a problem of relatively narrow scope. General modeling environments (e.g. 3-D groundwater flow codes) are not conducive to rapid student use because of the "expertise" problem. General modeling environments often offer so many options that students become lost making decisions about factors that do not apply to their work, and exceed the bounds of their expertise. Narrow focus codes have much of the problem statement impressed on the program structure so use is a relatively simple matter of selecting appropriate options.

Mechanistic Flexibility - Although packages should have a narrow focus, they should offer flexibility in specifying the environmental and engineering science expressed in the problem. It is important for students to accept responsibility for the problems they solve, and important that students learn about the impacts of the engineering decisions they make as they apply analysis.

User-Friendly Front End - The nature of the "front end" is of particular importance in educational use. Student users must be given a great deal of assistance with the structure of the input and clear explanations about the information requested. Students may be asked to supply data they do not understand or have never even heard of. It is important that they be given as much assistance as possible in starting an analysis. It is also important that this be done with the minimum amount of information necessary.

Blunder Control - It is important for the code to seek to protect itself from ill-conditioned input. This is always a desirable feature, but is of particular importance for software used in education. There may be a reasonable expectation that professional users will understand the problem well enough to enter credible data. This is not a reasonable expectation for student users.

Learning Curve - It is important that students be able to master a code in a reasonably short time period. One might expect professional users to undergo a period of training. Numerous short courses are offered specifically to train professional users about common software products. Unfortunately, in educational use, if it takes too long to begin using a code, students will allocate their time elsewhere. A long "learning curve" also dilutes the value of immediate impact as courses must move on to new subjects. When students cannot invest enough time to make a code work, they come away from the experience with very negative feelings about the value of environmental modeling.

Rapid Run Times - It is helpful if educational-use codes execute rapidly. This is not as important in professional use, but for educational use, immediate user feedback is very important. If a code is used in classroom lectures, it must yield results in seconds. If a code is used in interactive learning sessions outside of class, run times can be minutes, but generally not tens of minutes. One must be very cautious about using codes that require hours once "production" runs are launched. In professional use, production runs are usually successful. In the hands of students, production runs are often unsuccessful. Students can become trapped in iterative experimentation that challenges academic schedules (i.e. due dates) and leads to desperation and frustration. This can also consume computational resources (e.g. time on the department's PC's) that is required for other purposes and by other professors.

Visual Impact - It is very helpful if the software yields results in some visual form. This is a desirable feature of any code, but is even more important in educational applications because students seldom have as clear a vision of a problem as their professional counterparts. Developing the ability to visualize complex problems and solutions is an important part of education. In addition, if the code is for classroom instruction, the code must enable the instructor with something visually exciting. There is little less exciting than sitting in a classroom watching pages of numbers scroll by.

Detailed Result Documentation - It is important that the results of software analyses be presented with detailed documentation. Professionals are expected to understand the results and to be prepared to use them in some meaningful way. In education, the goal is often simply to understand the results. Students are expected to learn about the problem being modeled by examining the results, and they cannot do so unless the software provides rich, detailed information about how the final results were attained.

Processing Feedback - Dynamic feedback is also more important in educational use than in professional use. Professional users expect their analyses to be successful. Students do not know if the analyses they launched will actually run. It is very helpful if the code provides dynamic feedback on the state of the analyses as it runs. This may be inefficient, and may slow the execution of some codes, but is well worth the investment if it helps students remain engaged in the analysis process.

Low Cost - Cost is also more important in educational use than in professional use. Obviously, cost is important to everyone, but in professional use there is a better-defined revenue source to support cost. This is generally not the case in education. Although student enrollments can be equated with revenue, we seldom finance coursework this way. Educators can also be reluctant to commit to expensive proprietary computer packages because of commercial and legal implications. Experience has indicated that educators will easily make modest expenditures on software (< $500) but are reluctant to commit to more expensive codes unless they are general application tools with campus wide use implications.

Machine Flexibility - Hardware requirements can always be troublesome if they are unusual, but this is much more troublesome in educational applications than professional use. Machines (e.g. specific workstation platforms) dedicated to specific software products or modeling environments generally find few educational applications beyond the hands of their developers. It is simply too inefficient to dedicate educational computers to single software applications.

Technical Documentation - It is important that the conceptual design of software, and the solution algorithms applied be carefully documented. It is extremely important that students take responsibility for the function of software packages. If they think of software as a "black box" that just yields answers, the focus will be on the answers, and not on how these answers were created. Professional users may have mastered a subject, and be confident about the performance of a software package. Such users may only want answers. Students rarely satisfy either of these criteria. Students must focus on how and why a code works or they will learn little about the environmental phenomena being simulated. Without detailed technical documentation, students will not be able to take responsibility for the work they produce. Unfortunately, if you think that this kind of documentation is common, you are wrong. Often software developers consider this to be part of the "proprietary" portion of their product and provide scant details about the algorithmic function of their products.

Information-Rich Example Applications - It is also important to provide students with "rich" example simulations as starting points in their learning process. It is common that software products include examples. It is uncommon that "rich" examples are provided. "Rich" examples provide a detailed problem description and detailed explanations about how and why information is used in the simulation. This kind of example is avoided because software developers are concerned about the liability of coaching users on how to solve environmental problems. This is understandable from the software developers point of view, but "coaching users" is a reasonably good definition of environmental education. If information-rich examples are not included with the code, they must be created by the instructor. This is a fair requirement, but constrains the use of software by instructors who do not have the time to accomplish

this. One of the significant accomplishments of the Gateway consortium is the ability to share "rich" examples across multiple campuses.

It will probably not be possible to find all of these attributes in each piece of software, but it is worth considering what an ideal package should offer. Software developers should consider what could be added to their products to extend the market for code applications. Hopefully you will see elements of these attributes demonstrated and discussed in the following presentations.

Summary and Conclusion

Between May, 1995 and August 1997 collaborators from seven Gateway consortium universities worked to develop shared resource modules for Environmental Engineering education. This effort focused on four impact areas and resulted in approximately 24 modules. These modules have found applications within the Gateway consortium, and we hope they will find their way into applications across the variety of ways American universities express Environmental Engineering programs. Experience has demonstrated that quality modules are difficult to develop, and difficult to distribute to educational colleagues because of the nontraditional nature of the material. However, experience has also demonstrated that a successful effort will be well worth the work required.

Acknowledgement

This effort was supported by National Science Foundation Gateway Coalition grant NSF# CID-U-05-CW. The author also grateful the acknowledges the efforts of his colleagues and collaborators at the participating Gateway Universities.

References

Cataldo, J., "A Drainage Module for Environmental Engineering", Proceedings of the 1998 ASEE Annual Conference & Exposition, Seattle, WA, June 28 - July 1,1998, (in press).

Fentiman, A.W. and Jennings, A.A.,"Software Applications in Solid and Hazardous Waste", Proceedings of the 1998 ASEE Annual Conference & Exposition, Seattle, WA, June 28 - July 1,1998, (in press).

Flora, J.R.V. and McAnally,. A.S., "Treatment Plant Instructional Modules in Environmental Engineering", Proceedings of the 1998 ASEE Annual Conference & Exposition, Seattle, WA, June 28 - July 1,1998, (in press).

Jennings, A.A., "A Bioremediation Teaching Module Based on BIOlD", Environmental Modelling and Software, 12(l), l-18, 1997.

Jennings, A.A. and Kuhlman, S.J.,"An Air Pollution Transport Teaching Module Based on GAUSSIAN MODELS 1.1" , Environmental Modelling and Software, (in press), 1998.

Jennings, A.A., "A Vapor Extraction Teaching Module Based On AIRFLOW/SVE", Environmental Modelling and Software, (in press), 1998.

Mesania, F.A. and Jennings, A.A., "A Hydraulic Barrier Design Teaching Module Based on HELP 3.04 and HELP Model for Windows V2.05", Environmental Modelling and Software, (in press), 1998.

Roy, D., Jennings, A.A. and Maillacheruvu, K.,"Air Pollution Transport Teaching Modules", Proceedings of the 1998 ASEE Annual Conference & Exposition, Seattle, WA, June 28 - July 1,1998, (in press).

Biographical Information

Dr. Jennings is a Professor of Civil Engineering at Case Western Reserve University. He received his undergraduate degree from the Rochester Institute of Technology in 1975, his M.S. degree from the University of Massachusetts in 1977 and his Ph.D. degree from the University of Massachusetts in 1980. He taught at the University of Notre Dame and The University of Toledo prior to moving the Case Western Reserve University to head the program in Environmental Engineering. His undergraduate teaching responsibilities have included Fluid Mechanics, Hydraulic Engineering and Hydrology, Water Supply, Environmental Engineering Laboratory, Solid and Hazardous Waste and Water Resources Engineering. Dr. Jennings has also taught graduate courses in Subsurface Hydrology, Environmental Engineering Principles, Hazardous Waste Management, Advanced Groundwater Analysis, Environmental Engineering Modeling, Applied Groundwater Modeling and Environmental Remediation.

Dr. Jennings is an active researcher in the areas of groundwater flow and contamination, hazardous waste management, GeoEnvironmental engineering, environmental decision-making, soil remediation and urban hydraulics. He is a member of the American Society of Civil Engineers (ASCE), American Water Resources Association (AWRA), Association of Environmental Engineering Professors (AEEP), Soil Science Society of America (SSSA), American Geophysical Union (AGU), Water Environment Federation (WEF) and the Cousteau Society (Founding Member). He is also the Co-Editor-in-Chief of the international journal *Environmental Modelling and Software.*