Development of a Set of Closed Laboratories for an Undergraduate Computer Science Curriculum

William A. Wulf
University of Virginia
Charlottesville, VA 22901
(804) 982–2223; e-mail: wulf@cs.virginia.edu

DUE-9156112
FY 1992 $ 204,514
FY 1993 $ 200,334
FY 1994 $ 200,152
FY 1995 $ 64,548
Computer Science

As a discipline, computer science has seen many dramatic changes in its brief history. Through new textbooks and an evolving curriculum, the content of the undergraduate computer science education has, for the most part, kept pace with these changes. But its pedagogy has hardly changed. It emphasizes individual skill in writing, from scratch, small programs in an obsolete language. This emphasis is the antithesis of what is needed by contemporary computing professionals. This project is developing and evaluating a new undergraduate curriculum and supporting materials focused around the practice of computing, especially in the first two undergraduate years. This pedagogical shift has enormous potential for improving undergraduate computer science education.

The focus of the new curriculum is a core sequence of closed laboratories. Lecture content and materials support the laboratories instead of the reverse. While content can always be improved, quantum improvements cannot be expected. Laboratories, by contrast, have been largely neglected in this "laboratory science."

The Department of Computer Science is developing a "practice-centric" undergraduate computer science curriculum. Specifically, it is developing a collection of laboratory materials and exercises that support this approach. Also central to this approach is an increased degree of rigor in all courses, coupled with the early introduction and subsequent use of modern software development methods and tools.

In the new computer laboratory, furnished with state-of-the-art hardware and software, students work together through a series of exercises designed to encourage teamwork and brainstorming while reinforcing course content. These exercises are based on large hardware/software systems, incorporating actual industrial/corporate hardware and software products. The results of students' work are exhibited and critiqued by peers. The exercises also employ the case-study model to facilitate understanding of an entire system with fun equipment, such as sound generators and robot arms, that is available for laboratory projects. The use of advanced CASE tools also is part of laboratory exercises.

Each laboratory exercise, as well as the overall curriculum, is being evaluated using surveys, pretest/post-test, interviews, etc. Evaluation is done on a continuous basis for each set of closed laboratory exercises, allowing modification of the material over several groups of students. Questionnaires for graduates and their employers provide needed information about the long-term benefits of the revised undergraduate computer science curriculum. When the initial sequence of introductory courses has attained the stated objective, beta tests of the materials will be conducted at other schools. This closed laboratory approach provides students with more real world experience while learning the basic tenets of computer science.

"This-is-how-a-computer-works": A Software System for Lecture Demonstrations, Laboratory Exercises, and Home Study

Alan W. Biermann, Dietolf Ramm
Duke University
Durham, NC 27708
(919) 660 6500; e-mail: awb@cs.duke.edu

DUE-9455507
FY 1995 $ 100,000
Computer Science

This project continues ongoing work to create a simulator for an idealized computer that enables students to see its internal workings as it executes a computation. The student can observe the compilation process as a higher level language is translated into assembly language. Then he or she can watch the resulting assembly language executed on the architecture at the register level. Finally, the system can display one of the functional circuits (the adder) at the switching circuit level and one can watch switches turn and the electricity flow as the computation proceeds.

The system offers the user the option of calling for a tutorial description of the individual steps as they proceed. On the appropriate menu call, a little professor jumps onto the screen, and in cartoon style, describes the events of the computation. The system is designed to be used by a lecturer on a wide screen display during a class, by students in laboratory sessions, or as a home study aid. The system is being tested on a group of thirty students and their levels of achievement in learning how a computer works will be evaluated. It is available to any potential users via ftp and has been described in numerous papers and presentations to the community. The system should be a powerful aid in the teaching of the principles of operation of modern computers.

Integrating Social Impact and Ethics into the Computer Science Curriculum

C. Dianne Martin
George Washington University
Washington, DC 20037-2353
(202) 994-8238

DUE-9354626
FY 1994 $ 43,744
FY 1995 $ 55,000
FY 1996 $ 40,000
Computer Science

The purpose of this project is to develop a plan and materials for integrating social impact and ethics topics across the computer science curriculum. It addresses two major problems that hamper the implementation of an across-the-board curricular change: (1) the lack of materials that can be adapted or adopted into the existing computer science (CS) curriculum; and (2) the lack of awareness and expertise on the part of most CS faculty regarding the need and methodology for presenting such material in their courses. The project comprises three interrelated tasks: (1) defining the content of teaching modules to facilitate the presentation of these topics; (2) developing the actual modules to be disseminated as a teaching kit to interested CS departments and faculty members; and (3) developing a pilot faculty enhancement seminar to prepare interested CS faculty members to use the materials. The first task is to implement a two-day working conference of experts in ethics, social impact, and computer science curriculum to determine the appropriate content for the teaching modules. Topics for a first-year computers-and-society course, the traditional computer science core courses, and the senior design course are being discussed. The anticipated outcome of this conference is a consensus regarding which topics, how much time, and the level of such topics that should be included in the CS curriculum. The second

task is to implement the teaching modules that result from the working conference. These modules include scenarios, exercises for discussion and written comment, and teaching strategies for presenting the topics. The modules are being developed for a first-year course, for computer science core courses, and for a senior design course. These modules will be sent out for comment and pilot testing by computer science professors. A kit of teaching modules is being developed for general dissemination. The third task is to implement a pilot two-day faculty enhancement seminar to provide guidance to CS professors regarding effective use of the materials. Based upon results from the pilot seminar, a kit for faculty enhancement seminars will also be developed for general dissemination.

Educating the Next Generation of Information Specialists, in Collaboration with Industry

Michael C. Mulder, Doris K. Lidtke
University of Southwestern Louisiana
Lafayette, LA 70503-2701
(318) 231-6000

DUE-9455450
FY 1995 $ 161,850
FY 1996 $ 151,463
FY 1997 $ 153,682
Computer Science

This work produces a greatly needed and major revision (or complete revamping) of Computer Information Science (CIS) oriented curricula. Specifically it provides detailed curriculum guidelines, supporting laboratory material, original learning/teaching paradigms, and a methodology for dealing with complex information systems. This major component of the computing sciences has been overlooked by practitioners, educators, and the professional societies for at least 15 years. During this same period the need for complex, large grain information systems to support all aspects of business, industry, and government has exploded; similarly, the technologies supporting the generation and usage of information have changed dramatically. This has led to the current dilemma: the availability of exciting new information technologies to be applied, but few new hires or practitioners, if any, properly prepared to effectively apply these advances. This academe/industry collaborative project addresses this dilemma, and will produce solutions. The work builds on what is available today in CIS, adding to it substantial new course and laboratory material, a methodology for dealing with large grain information system complexity, development of teamwork and communications skills in students, and embeds problem-solving and design skills within the recommended curriculum. To support this curriculum, appropriate materials are developed and tested at selected university/industry test sites. A new learning/teaching paradigm (i.e., Just in Time Learning) is developed and tested, in which qualified students (guided by a faculty member) are electronically linked with an ongoing industry large grained information systems project. There are participating project team members, receiving both project experiences and Just in Time classroom/laboratory experiences. The Boeing Company and Motorola are committed to supporting this original learning/teaching paradigm. Three eminent CIS specialists form the National Visiting Committee to critique and guide the project, and 50+ additional external specialists will review the results of the project. In this manner, the quality and applicability of project results are assured. Widespread and timely dissemination to the computing profession will be conducted.

An Interactive Laboratory Infrastructure for Computer Science

Rockford J. Ross
Montana State University
Bozeman, MT 59717
(406) 994-0211; e-mail: ross@cs.montana.edu

DUE-9455588
FY 1995 $ 93,344
Computer Science

The objective of this project is the design and development of an infrastructure around which formal laboratories in computer science can be implemented. The central component of the infrastructure is a software system called DYNALAB, for DYNAmic LABoratory. DYNALAB currently runs on IBM-compatible personal computers under Microsoft Windows (with the Win32 extension) or Windows NT, as well as UNIX-based XWindows systems. Students using DYNALAB have access—through a user interface oriented towards complete novices—to a comprehensive, easily modifiable, and extendible library of programs and experiments, including experiments in program structure (e.g., iteration, selection, recursion, execution-path determination, parameter passing mechanisms, functions, and procedures), black box determination of algorithms, time complexity, space complexity, program verification, and others. A sophisticated program animation component allows the execution of programs both forward and in reverse, while displaying dynamically and in interactive fashion the pertinent aspects of program execution, such as the currently executing statement, values of variables, pointer references, and statement and memory cell counts (for time and space complexity experiments). An algorithm animation component also allows abstract graphical representations of algorithms to be presented on a screen in forward and reverse modes for algorithm and data structure studies. Eventually, concept animations will be included for the study of fundamental computer science concepts, such as compiling, multitasking, and intractability. The desirability of formal laboratories in the computer science curriculum is well-documented, but few institutions have incorporated them for lack of laboratory resources. DYNALAB represents one such resource. DYNALAB will be distributed as an ancillary to textbooks in programming, as well as data structures and algorithms. It will also be available on the Internet, following links beginning with http://www.cs.montana.edu/~ross. It is expected that DYNALAB will be widely adopted as a tool around which computer science laboratories can be designed.

Program Derivation for a Data Structures Course

David A. Naumann, Richard T. Denman
Southwestern University
Georgetown, TX 78626
(512) 863-6511; e-mail: naumann@ralph.txswu.edu

DUE-9455660
FY 1995 $ $23,698
Computer Science

Computer programming is often taught primarily by example, and there is widespread interest in more systematic methods, especially mathematically rigorous "formal methods." There are now undergraduate level textbooks and teaching materials for beginning programming courses using the method called program derivation, whereby correct programs are derived from their specifications by calculation. The objective is to develop lecture notes and programming assignments for an undergraduate data structures course based on program derivation. The materials are based on materials now being used in the classroom, and will be evaluated by educators at other universities for content and suitability for use in undergraduate courses in data

structures. By meeting the need for instructional material based on the extensive research literature on program derivation, this project will directly benefit educators who wish to teach scientifically-based systematic methods of programming. By providing material on par with textbooks for the more traditional approach, this project will help to provide a basis for meaningful comparison between curricula based on formal methods and other curricula.

Cognitive-Based Approach to Introductory Computer Science Courses

David L. Feinstein, Herbert E. Longenecker,
David D. Langan, Michael V. Doran
University of South Alabama
Mobile, AL 36688
(205) 460-6101; e-mail: feinstein@cis.usouthal.edu

DUE-9455522
FY 1995 $ 89,976
Computer Science

A cognitive-based approach is used to develop comprehensive materials for the first courses in computer science based on Implementation D of Computing Curricula 1991. The distinguishing features are: (1) materials based on a strategic sequencing and the associated level of mastery of key topics, (2) a topical coverage carefully based on a spiral approach to information presentation, (3) an integral use of structured labs as a necessary component, (4) an emphasis on frequent feedback to facilitate learning and evaluate the effectiveness of instruction, (5) remediation materials for students requiring additional assistance to meet the required levels of mastery, (6) an early use of teams, (7) a student surveying tool used to track all students to provide outcome assessment, and (8) review and evaluation by multiple institutions for iterative material refinement and national dissemination. The project is being carefully evaluated by a team of eight consultants with expertise in the fields of computing, computing education and educational psychology. The materials developed are also being critiqued by professionals at a wide variety of other institutions for their input as to the applicability of the materials to students at their institutions. The resulting materials will be made available in hard copy and through electronic media.

Pattern-Based Programming Instruction

J.P. East, Walter E. Beck, Janet M. Drake,
Sarah R. Thomas, Vernon E. Wallingford
University of Northern Iowa
Cedar Falls, IA 50614-4800
(319) 273-2311; e-mail: east@cs.uni.edu

DUE-9455736
FY 1995 $ 69,812
Computer Science

A detailed course outline for a pattern-based introduction to computer programming is being developed. The course begins with a statement of theory and principle that both explains and guides the rest of the work. Common types of computing problems that provide coverage of typical programming course content are being identified. Next, one or two examples for each problem type that will be used to illustrate the basic algorithmic pattern are chosen. A number of alternate problems requiring modifications of each basic pattern is being identified and the course outline is being finalized.

Three activities are being used to assess the course: (1) Faculty perceptions as to effectiveness, coverage of material, ease of use, student progress, etc., is being noted. (2) A questionnaire is being developed and administered to address student perceptions of their own understanding and their comfort with the approach. (3) Questions common to final examinations for various programming courses are being compiled and the effectiveness of this approach to that of traditional instruction is being compared. After assessment and revision as called for, at least one additional course will be offered using the same approach, but perhaps a different language.

Papers and presentations will be produced and submi tted to appropriate computer science education journals and conferences. The theory and principles involved in the approach, as well as descriptions of the curriculum, will be addressed.

The instructional material will provide a solid testbed for further research and explication of standard problems and programs in computing. The pattern-based approach should significantly improve the quality of programs produced by students after a first course. While this project does not focus on any special audience, its approach to programming instruction should ultimately enhance both attraction and retention of all students, including women and minorities.

A Second Course in Computing for Non-Majors

Jerry Waxman
CUNY Queens College
Flushing, NY 11367-1575
(212) 520-7000; e-mail: waxman@qcvaxa.acc.qc.edu

DUE-9455635
FY 1995 $ 85,835
Computer Science

Information will be the coin of the realm in the 21st century, and learning to locate, evaluate, utilize, and disseminate it will be at the very center of the university's mandate. In the past, only some science and engineering students needed extensive high level computing skills; however now, and increasingly in the future, fluency with computational environments will be a decisive factor in both academic and commercial success. As a sequel to the PI's highly successful first course for non-computer majors, sponsored jointly by NSF and the Department of Education, this second course, which focuses on the acquisition and presentation of data and information, is being developed.

For data acquisition, the course gives the students high level mastery of the Internet. For the presentation of information, students master a multimedia authoring system. The Internet segment of the course focuses on the following topics: (1) what is the Internet and what is it not; (2) resources available on the Internet for teaching and research in various fields; (3) where to locate additional information; (4) navigating the Internet via Gopher, Archie, Veronica, WAIS and WWW, e-mail, FTP, Telenet and List servers; and (5) using USENET, setting up news groups and IRC on an Internet node set up and maintained at the College.

The multimedia segment of the course uses the Toolbook environment from Asymetrix Corporation and focuses on how to construct simple "books" with hot keys and buttons, how to create multipage branching books, how to create motion and animation, how to design and create hypertext systems, how to design and create interactive multimedia, and how to program sophisticated multimedia applications within Toolbook.

Computer-Based Laboratories for Introductory Computing Courses for Science and Engineering Majors

Joseph L. Zachary
University of Utah
Salt Lake City, UT 84112-1201
(801) 581-7200; e-mail: zachary@cs.utah.edu

DUE-9455478
FY 1995 $ 87,125
Computer Science

In traditional science and engineering degree programs, computation is commonly treated as a problem-solving tool best studied and applied in isolation from "real" science and engineering. Students are typically required to take a programming class to learn how to program, and perhaps a numerical analysis class to learn how to apply their programming skills. This traditional approach ignores the fact that computation has evolved into an essential way of illuminating scientific and engineering principles. Researchers now choose among the theoretical, experimental, and computational approaches to studying scientific phenomena. It has become increasingly clear that computation should be exploited in science and engineering education for its descriptive and analytical powers. In the past, the barriers to using computational techniques were high. The emergence of application programs such as *Maple* and *Mathematica* has greatly lowered these barriers. The computational power made available per unit of student effort is orders of magnitude higher for these packages than it is for traditional programming languages. This project is developing a suite of computer-based laboratories suitable to support a two-semester freshman/sophomore-level computing course for science and engineering majors. Each laboratory illustrates the computational solution of a typical problem from a field of science or engineering, builds upon freshman-level concepts from mathematics and physics, and leads the student through the process of learning the computing technology (including mathematics packages and conventional languages) required to solve the problem. Workshops will be conducted annually to train undergraduate faculty in the use of the course materials.

A Hypermedia Lab Manual for Operating Systems

Stephen J. Hartley
Drexel University
Philadelphia, PA 19104
(215) 895-2678; e-mail: shartley@mcs.drexel.edu

DUE-9455307
FY 1995 $ 19,428
Computer Science

Sending students to a local copy center to get copies of class handouts is an inefficient, high cost method of distributing class handouts. In addition, the full impact of algorithm animations such as the dining philosophers cannot be captured in snapshots on paper. What is needed is a method of making handouts and documents available to students that does not use paper, does not cost money, is not limited to plain text, contains more than just algorithm animation snapshots, and facilitates ease of access throughout the university network. The goal of this project is to transform some operating systems and concurrent programming handouts into a hypermedia laboratory manual residing in files accessible by Drexel students from any network location. The files are written in Hypertext Markup Language (HTML) and contain hyperlinks to program source code, screen snapshots, and movies of algorithm animations in MPEG or QuickTime format. Students use a World Wide Web browser program such as Mosaic to access

the lab manual via Hypertext Transmission Protocol. Since the Drexel dormitories are now on the network, students are able to access the lab manual from their rooms using their Macintoshes. Feedback will be solicited on the ease of use and effectiveness of the hypermedia lab manual from Drexel students as well as from other students and faculty accessing the lab manual over the Internet. The hypermedia lab manual will be made available over the Internet to operating systems and concurrent programming students and faculty at other institutions. Such availability will be announced in several relevant USENET newsgroups. A description of the project and its results will be submitted for publication and presentation at a conference. This is an innovative use of the university computers and network to make handouts and other documents available to students in an efficient, low cost manner. Students and faculty at other institutions on the Internet will also be able to access this hypermedia operating systems lab manual.

An Assertion-Based Programming Methodology for Introductory Programming Courses

Jerud J. Mead, Anil M. Shende
Bucknell University
Lewisburg, PA 17837
(717) 523-1271; e-mail: mead@bucknell.edu

DUE-9455489
FY 1995 $ 58,283
Computer Science

A perusal of current introductory programming texts shows that while the syntax of structured statements is described in formal terms, the semantics of structured statements is described in informal verbal terms. The result is that students leave introductory programming courses with a good understanding of the syntactical aspects of structured programming and a poor understanding of the semantical aspects. The programmers will have difficulty assessing either formally or informally the correctness of software which may control critical computer systems. The goal of this project is to develop and assess a programming methodology for the introductory programming sequence, based on a blend of operational and axiomatic semantics, which integrates semantic content (in terms of assertions) into the programming process.

A textbook which presents the methodology is being written and assessed. The text is language-independent and useful as either a stand-alone or a supplement to an existing introductory programming text. The impact of this project is on four groups. (1) Instructors of introductory programming courses are able to integrate the semantic-based methodology into their existing courses without changing the primary text: the supplementary text provides them formal support for integrating the methodology. (2) Students learning the methodology via the supplementary text gain more thorough understanding of the program semantics because the semantics are presented formally in terms of assertions. They are better equipped to convince themselves and others that their programs are correct because their programs carry the evidence. (3) Students, instructors, and computing professionals who have not had the advantage of a semantic-based programming methodology will be able to use the text to familiarize themselves with the more formal semantic side of programming. (4) Finally, the public will be long-term beneficiaries of the project.

The GeoSim Interface Library for Introductory Programming Courses

Clifford A. Shaffer, N. D. Barnette

DUE-9455403

Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
(703) 961-6000; e-mail: shaffer@vt.edu

FY 1995 $ 19,768
Computer Science

The GeoSim Interface Library (GIL) provides a common set of Graphical User Interface (GUI) functions for programs running under the MS-DOS, Macintosh and XWindows programming environments. Application program software is written to access GIL. In this way, programs need only be written once to run under these three programming environments. GIL handles most typical GUI tasks, such as window display; backing store management for overlapping windows; and mouse interaction with buttons, drag regions, lists and sliders. GIL supports "help screens" and "alert boxes." GIL also includes additional support for non-typical applications development features such as log files, data file processing, and time-driven functions independent of user interface objects and actions (used for discrete event simulations). Existing GUI development systems do not provide the multi-platform capability necessary for our diverse student population. Fortunately, GIL has proven to be easy to use by students and researchers alike, and can easily be adapted to class use. This project seeks to obtain the modest manpower resources necessary to make GIL a tool that can be used by undergraduates in introductory programming courses. GIL provides a GUI library for use in writing class programming assignments. Use of a GUI in introductory programming classes allows students to begin early practice in writing applications programs more like what they will do in their jobs. Enhancements to the GIL system include a Graphical Interface Developer that allows students to interactively place user interface elements on the screen, better support for fonts and window backing store, and classroom-ready documentation and examples. GIL, including source code and documentation, will be freely distributed via tabs Internet.

Development of Lecture Room Demonstration Experiments to Improve the Computer Literacy Course

David K. Walker, Hamid Chahryar
Marshall University
Huntington, WV 25728-2947
(304) 696-3170; e-mail: walker@muvms6.wvnet.edu

DUE-9455314
FY 1995 $ 59,121
Computer Science

Lecture room demonstration experiments, which actually show how things work, are relatively new to computer science educators. In most other fields of science, demonstrations have proven quite effective in combination with a variety of teaching methods and have been successfully used for many years. Such "live" demonstrations, especially for introductory classes, awaken a keen interest in the "chalk and talk" lecture and tend to be remembered long afterward. Lecture room demonstrations need to be considered seriously as a teaching aid for helping to revitalize introductory computer science courses. The real problem is that very few instructional materials are currently available on this subject. This project addresses the need to develop computer science lecture room demonstration materials. More specifically, it concentrates on development of eight innovative, low cost, computer hardware-related lecture room demonstration experiments to improve the computer literacy course taken by non-computer science majors. These demonstrations help students to better understand basic concepts of how computers operate and permits them to see firsthand, a number of attention-grabbing, real-world applications. Students learn that computers are used for more than just running software applications. The

demonstrations show students how computers and peripherals work; illustrate aspects of telecommunications and networking; show actual business and industrial applications; and present emerging computer-related technologies such as multimedia, hypermedia, and virtual reality. Details of how to build and set up the demonstration apparatus, teaching techniques, and results from classroom trials are made readily available via pamphlets, presentation at workshops, and the Internet. This project is expected to be of great interest and benefit to all teachers of the computer literacy course.

Using Iconic Environments to Teach Procedural and Object-Oriented Programming Concepts

Donald J. Bagert, Ben A. Calloni
Texas Tech University
Lubbock, TX 79409-4609
(806) 742-2011; e-mail: bagert@cs.coe.ttu.edu

DUE-9455614
FY 1995 $ 56,999
Computer Science

This project addresses the question: Can icon-based programming languages be used to teach first-year programming concepts to undergraduate students more effectively than text-based languages? The specific objectives are: (1) To develop a complete set of course materials for BACCII, an already-existing iconic programming environment under ongoing development at Texas Tech University, and use them in the first two computer programming courses. For this project, the first course is an introduction to procedural programming, software engineering, and C++, while the second course teaches concepts in object-oriented programming and data structures, also in C++. (2) To determine whether iconic environments (and BACCII in particular) can be used to more effectively teach procedural and object-oriented programming concepts in the first two courses, while still allowing students to learn the syntax of text-based languages such as C++. This project is testing the course materials at Texas Tech, while a possible later project would allow for pilot programs to be started at five higher education institutions. Data will be collected and evaluated not only for the general student population of each course, but for various groups such as women, underrepresented minorities, and future teachers. (3) To advance research in the area of using iconic programming languages as learning environments. The results of the data from Texas Tech and the other five pilot programs will allow for an evaluation of how future iconic languages for learning programming skills should be developed. If it can be demonstrated that iconic languages perform significantly better text-based languages in the learning of programming skills for a wide variety of schools and student groups, it could revolutionize how software development techniques are taught.

Technology in Teaching and Research: A Look at Private Education in Appalachia

Alice W. Brown, Roy Vignes
Appalachian College Association
Berea, KY 40403

DUE-9554456
FY 1995 $ 19,166
Computer Science

This ambitious, long-range project is designed to unite the efforts of faculty in private colleges across the Appalachian region to strengthen their use of technology in the undergraduate curriculum. The first step in the project, "A Look at Private Education in Appalachia," is to determine what technology is currently in place and how faculty are using it in their classrooms, both for instruction and research. The study is expected to show that while many of the Appalachian colleges and faculty are seriously lacking in the resources necessary to employ technology to strengthen their instruction, in far more cases the colleges and their faculty are on the edge of excellence in their access to and use of technology. At the end of the study, a college or faculty member will be able to examine itself or himself in view of what is happening at like institutions. The college will be able to determine what is necessary to strengthen its competitive edge in recruiting and maintaining students, and the faculty will be able to identify other faculty across the region who share their interest or who can strengthen their knowledge and skills. The final report documenting the results of the study will be available for dissemination to colleges across the nation who may also want to prepare for the 21st century.