IDEAS

Instructional Design-Based

(Curriculum)

Analysis and Synthesis

A Preliminary Proposal to the Applications of Advanced Technologies Program of the National Science Foundation

Primary Contact: Roger B. Dannenberg

School of Computer Science (SCS)
Carnegie Mellon University
Pittsburgh, PA 15213

internet: rbd@cs.cmu.edu
fax: 412 621-6787
phone: 412 268-3827

A. Introduction to IDEAS

We propose to produce an Instructional DEsign-based Analysis and Synthesis environment (IDEAS). The fundamental distinguishing feature of IDEAS is that IDEAS operates at the level of curriculum development. IDEAS is a software architecture for curriculum design in precisely the same sense the spreadsheet is a software architecture for financial analysis and management. IDEAS is suitable for designing curricula for traditional courses, but its power is most crisply seen in the development of new courses and training modules that parsimoniously merge two or more traditional fields or courses of study. Other educational systems, rather than being competitors for IDEAS, can be incorporated into the construction of IDEAS curricula.

Our country spends even more on education than it does on health care, yet education remains a very human-intensive practice based for the most part on technology of a previous century. It is true that recordings, film, and television have much enriched the educational process, but these contributions come in pre-packaged forms over which the teacher has little control. Workers and managers in other fields of everyday activity benefit from the wide range of computer software now available for writing and printing (word processors, various printing devices), scheduling (calendars, project tracking software), financial analysis and management (spreadsheets), communication (electronic mail, fax software), and archiving (file systems, search and retrieval software), to name a few. These developments were of course very much motivated and supported by the needs of business, and integrated "office packages" putting all this functionality together in a unified system are now popular. Today, most of modern business is unthinkable without the use of the electronic medium. Though computers are entering schools and colleges in ever increasing numbers, and advantage can be taken of off-the-shelf software, the integration achieved by business is not apparent in education. Much, much more has to be done to produce a comprehensive software architecture for supporting the work of educators, authors, teachers, and students.

There are, to be sure, many computerized teaching environments in service and in development. A partial listing includes: intelligent tutoring systems; text-based drill and practice systems; speech recognition and synthesis systems for language training; complex simulation environments making sophisticated use of multimedia; design systems that let teachers and learners collaborate with computer support. There are also very useful software packages—too numerous to mention—for drawing and various kinds of computation that have strong educational impact. Nevertheless, we do not feel that the generally narrow focus of these systems makes it at all easy for instructors to analyze, design, customize, and deliver an educationally sound *curriculum*.. It is, therefore, the purpose of our proposal to produce IDEAS, a new Instructional DEsign-based Analysis and Synthesis environment. IDEAS will operate at the level of curriculum development. It is capable of placing one or more other systems into an IDEAS curriculum, using them to help in the preparation and delivery of specific lessons. Moreover, by casting lesson materials in a common curricular context, IDEAS will provide a framework for analyzing the effectiveness of teaching strategies and whole genres of machine-assisted education. Our primary claims are that (1) IDEAS will support the rapid development of educationally sound curricula, and (2) IDEAS will effectively deliver individually tailored instruction to students. We cannot claim that IDEAS will be the dominant software architecture for education, but what we will do is offer a highly novel first approximation that will be open to further evaluation and expansion. To test these claims, we will make extensive experiments in three different domains of study. Therefore there are four major components to this project; the production of the IDEAS system and its use in three domains.

I. Why Is This Work Important?

One of the most difficult parts of designing educational material is the curriculum design. Hundreds or even thousands of concepts and explanations must be organized into a logical progression. Interactive teaching systems are even more difficult to design because they must actively adapt the curriculum to the individual student, selecting material based on the student's progress. There have been a few outstanding successes in interactive teaching systems, but the best of these systems have been hand-crafted by teams of PhD-level researchers. If new technology could reduce the time and complexity of design, it could foster an enormous increase in the amount of computer-based educational material, and dramatically lower the cost.

II. Where Is The Innovation?

The IDEAS approach was begun in response to an instructional designer's difficulty in producing a sound curriculum for a computer-based tutor. We discovered that applying basic tenets of instructional design in the formal setting of an intelligent tutor was quite difficult. Our solution to this problem was to automatically check a curriculum design for consistency and to use formal analysis techniques to aid the designer. Just one example is a check that each prerequisite to a unit of instruction is taught by at least one other unit. Since the relationships between units of instruction, the skills they teach, and their prerequisites are all explicit, it becomes possible to automatically select and deliver units of instruction according to an individual student's progress. Thus, IDEAS is innovative in guiding the designer to create a complete and sound curriculum, and in providing an automated, generic lesson selection engine for computer-based instruction.

III. Why Is New Technology Necessary?

In our experience, curriculum design is a very error-prone task. With traditional instruction, human teachers are flexible enough to overcome serious design problems. With computer-based tutors, however, poor design and design errors are not tolerable. Good curriculum design at low cost will require new techniques. This will allow more tutors to be developed and the potential for computer-based instruction, distance learning, and learning-on-demand systems to be realized.

IV. What Tests Will Be Performed?

Our goal is to show that IDEAS will support the rapid development of educationally sound curricula, and that IDEAS will effectively deliver individually tailored instruction to students. To test these hypotheses, we must incorporate the IDEAS technology into a curriculum design system. To show that IDEAS supports the rapid development of educationally sound curricula, we must use it to develop some curricula. We will start with existing informal curricula and compare them to curricula developed with IDEAS. We will also measure the time it takes to produce an IDEAS curriculum, monitor how well IDEAS gives feedback to the designer, and measure how long it takes to learn to use IDEAS.

To show that IDEAS can effectively deliver individually tailored instruction, we will use IDEAS to implement an intelligent tutor. One measure of how well instruction is personalized is to compare the lesson sequence seen by each of a set of students. Differences indicate that choices are being made on the basis of student differences. We can also compare student performance with automated lesson selection vs. performance with a fixed sequence of lessons is presented. Finally, we can compare students taught by an IDEAS-based tutor to students taught in a standard classroom.

It should be noted that there are two possible directions for evaluation. One is to evaluate IDEAS in terms of its support fro curriculum design and delivery. The other is to evaluate the effectiveness of a resulting curriculum or instructional system. Our plan is to emphasize the former so that we can provide a direct assessment of IDEAS, possibly leading to refinements. When we demonstrate that IDEAS works, we can then assess the impact of good curriculum design on the quality of educational material produced with IDEAS.

V. What Will Be Learned?

We will learn whether the IDEAS approach works and why. By building and using an IDEAS-based design system and an IDEAS-based tutor, we will also learn what is needed to use IDEAS in practice.

VI. How Generalizeable Is IDEAS?

One of the biggest attractions of IDEAS is that it is so domain-independent. The IDEAS approach was invented for use in a tutor for teaching piano, yet it seems even more natural for technical subjects. Because IDEAS is domain-independent, it should greatly reduce the cost of developing new tutors. This is because every IDEAS-based tutor will share the same design system and the same lesson selection engine.

VII. Benefits of This Research.

IDEAS is an innovative technology that has a potential for high impact at a national level. If IDEAS can lower the cost and raise the quality of computer-based intelligent tutors, then it will dramatically change how we teach many subjects. This change could significantly improve teaching and learning by offering massive amounts of effective learning material in an on-line, self-paced format. Specifically, this research will explore the new paradigm of computer-assisted curriculum design, determine its strengths and weaknesses, and lay the research foundation for practical implementations of IDEAS.

B. Research Overview

We propose to investigate IDEAS as a new paradigm for instructional design and delivery. The research will consist of the implementation of IDEAS as a computer-aided design system for education. This implementation will then be tested by applying it to the design and delivery of several courses.

I. Automated Analysis of Instructional Design and Delivery.

The Instructional Design-Based Analysis and Synthesis (IDEAS) environment is a solution to the difficult problem of designing a sound curriculum and adaptively instructing students. IDEAS is a domain-independent, technology-independent system that supports teaching and learning. IDEAS is particularly well-suited to computer-mediated, network-delivered education and training. IDEAS includes:

1. **Computerized curriculum analysis.** The field of Instructional Design offers widely-accepted advice for curriculum designers, but carrying out this advice by hand is tedious if not impossible. IDEAS reformulates and automates elements of Instructional Design, in effect providing a powerful computer-aided design (CAD) system for education.
2. **Dynamic, individually tailored instruction and "just-in-time" learning.** IDEAS can automatically select lessons to achieve an educational objective. If the student fails to learn a skill, an alternate lesson or sequence of lessons is selected automatically. Lessons from multiple courses can be integrated dynamically. This is useful in forming tightly focused remediation and in integrating material to make entirely new courses.
3. **A framework for sensibly merging diverse instruction approaches.** Model-tracing tutors, drill and practice, micro worlds, and multimedia are all important approaches to education. IDEAS does not force the curriculum designer to adopt a single modality of instruction. Radically different technologies are woven into a coherent, unifying instructional fabric.
4. **A platform for more tightly controlled educational experimentation.** IDEAS yields a formal, highly visual curriculum representation. This provides context for empirical evaluation of specific lessons and educational methodologies.

II. New Curriculum in Modern Math.

The Modern Math Curriculum makes innovative use of symbolic computation to engage the student in active learning. Lessons initially will be delivered in a computational lab-based course and later retargeted for self study and distance learning.

III. Distance Learning/Multimedia Course in Computer Science.

An existing course in data structures will be redesigned with IDEAS for independent study and distance learning. Interactive multimedia will be used for instruction. IDEAS will support curriculum evolution as teachers customize and personalize the course over time. This course was selected because it is a standard course. The point is to see if the approach does give leverage to the curriculum designer.

IV. Training in State-of-the-Art Light Microscopy.

Carnegie Mellon's Center for Light Microscope Imaging and Biotechnology is among the world's leaders in creating light microscopy technology and applications of that technology.

Currently, diffusion of these innovations is very slow. This project will facilitate a dramatic speedup of knowledge transfer through a distance-learning, IDEAS-supported curriculum. We believe training is the best example of IDEAS because it forces a blending of information sources. Light microscopy, for example, merges elements of optics, biochemistry, and computer graphics.

C. Background

Education and training are poised to become fundamentally and positively changed by the upsurge of interest in the National Information Infrastructure. The maturation of some genres of computer mediated learning, such as intelligent tutoring and constructivism, are encouraging. The reality of affordable, interactive multimedia computing is another exciting development. One thing that is lacking, however, is a practical approach to the mass-production of high-quality, low-cost, educational software.

A key to better educational software and other materials is good curriculum design. One solution is Instructional Design, which calls for integrity among instructional objectives, tasks that learners must perform, and the evaluation of task performance. Unfortunately, Instructional Design is tedious, expensive, and not understood by many educators.

IDEAS is based on Instructional Design, but overcomes these problems in several ways. First, IDEAS offers formal and specific design rules in place of the platitudes of Instructional Design. Specifically, IDEAS uses an explicit curriculum representation where skills, lessons, and their relationships are spelled out in detail. The design rules require that all skills and lessons are reachable and useful, that there are no circular dependencies, and that skills are distinguishable. Second, these design rules are checked quickly and automatically, eliminating much of the tedium and cost of using the Instructional Design approach. Third, the IDEAS environment guides teachers to apply Instructional Design techniques. Thus, IDEAS uses automation to lower both the intellectual barriers and the cost of good design. IDEAS reinforces good, systematic design practice.

This last idea is very important. Consider how a spreadsheet program encourages users to create a model with numerical constraints and then experiment with changes to parameters. By their nature, spreadsheets suggest this methodology. In a similar fashion, IDEAS will suggest and enable good design practice to educators who use it. Through templates that help to organize a curriculum and design rules that capture good design practice, IDEAS can help educators manage a detailed design. A key feature of many of the best computer-mediated learning systems is that lessons are small, sharply focused, numerous, and dynamically reconfigurable to the needs of the particular learner. IDEAS is especially well suited for assisting the curriculum designer in creating sound designs of this type.

Fundamental to IDEAS is that prerequisites, learning objectives, and what the learner must do in order to demonstrate success must be articulated clearly in behavioral terms. Progress is gauged by behavior indicating that a concept has been understood or a skill has been mastered. Because IDEAS makes explicit the relationship between empirical objectives, lessons, and prerequisites, the learner becomes the center of instructional design.

IDEAS uses a graph structure to represent lessons, skills, and their interdependencies (see Figure 1). Graph algorithms are employed for curriculum analysis and consistency checks. In addition to their role in analysis, IDEAS algorithms can merge two independent curriculum designs. For example, a course in organic chemistry and a course in biology could be automatically merged to

Currently, diffusion of these innovations is very slow. This project will facilitate a dramatic speedup of knowledge transfer through a distance-learning, IDEAS-supported curriculum. We believe training is the best example of IDEAS because it forces a blending of information sources. Light microscopy, for example, merges elements of optics, biochemistry, and computer graphics.

C. Background

Education and training are poised to become fundamentally and positively changed by the upsurge of interest in the National Information Infrastructure. The maturation of some genres of computer mediated learning, such as intelligent tutoring and constructivism, are encouraging. The reality of affordable, interactive multimedia computing is another exciting development. One thing that is lacking, however, is a practical approach to the mass-production of high-quality, low-cost, educational software.

A key to better educational software and other materials is good curriculum design. One solution is Instructional Design, which calls for integrity among instructional objectives, tasks that learners must perform, and the evaluation of task performance. Unfortunately, Instructional Design is tedious, expensive, and not understood by many educators.

IDEAS is based on Instructional Design, but overcomes these problems in several ways. First, IDEAS offers formal and specific design rules in place of the platitudes of Instructional Design. Specifically, IDEAS uses an explicit curriculum representation where skills, lessons, and their relationships are spelled out in detail. The design rules require that all skills and lessons are reachable and useful, that there are no circular dependencies, and that skills are distinguishable. Second, these design rules are checked quickly and automatically, eliminating much of the tedium and cost of using the Instructional Design approach. Third, the IDEAS environment guides teachers to apply Instructional Design techniques. Thus, IDEAS uses automation to lower both the intellectual barriers and the cost of good design. IDEAS reinforces good, systematic design practice.

This last idea is very important. Consider how a spreadsheet program encourages users to create a model with numerical constraints and then experiment with changes to parameters. By their nature, spreadsheets suggest this methodology. In a similar fashion, IDEAS will suggest and enable good design practice to educators who use it. Through templates that help to organize a curriculum and design rules that capture good design practice, IDEAS can help educators manage a detailed design. A key feature of many of the best computer-mediated learning systems is that lessons are small, sharply focused, numerous, and dynamically reconfigurable to the needs of the particular learner. IDEAS is especially well suited for assisting the curriculum designer in creating sound designs of this type.

Fundamental to IDEAS is that prerequisites, learning objectives, and what the learner must do in order to demonstrate success must be articulated clearly in behavioral terms. Progress is gauged by behavior indicating that a concept has been understood or a skill has been mastered. Because IDEAS makes explicit the relationship between empirical objectives, lessons, and prerequisites, the learner becomes the center of instructional design.

IDEAS uses a graph structure to represent lessons, skills, and their interdependencies (see Figure 1). Graph algorithms are employed for curriculum analysis and consistency checks. In addition to their role in analysis, IDEAS algorithms can merge two independent curriculum designs. For example, a course in organic chemistry and a course in biology could be automatically merged to

produce a single course in which organic chemistry concepts are introduced as needed to support the biology. Graph algorithms are also used for curriculum synthesis: a lesson selection engine searches the curriculum representation for the most appropriate next lesson for a student.

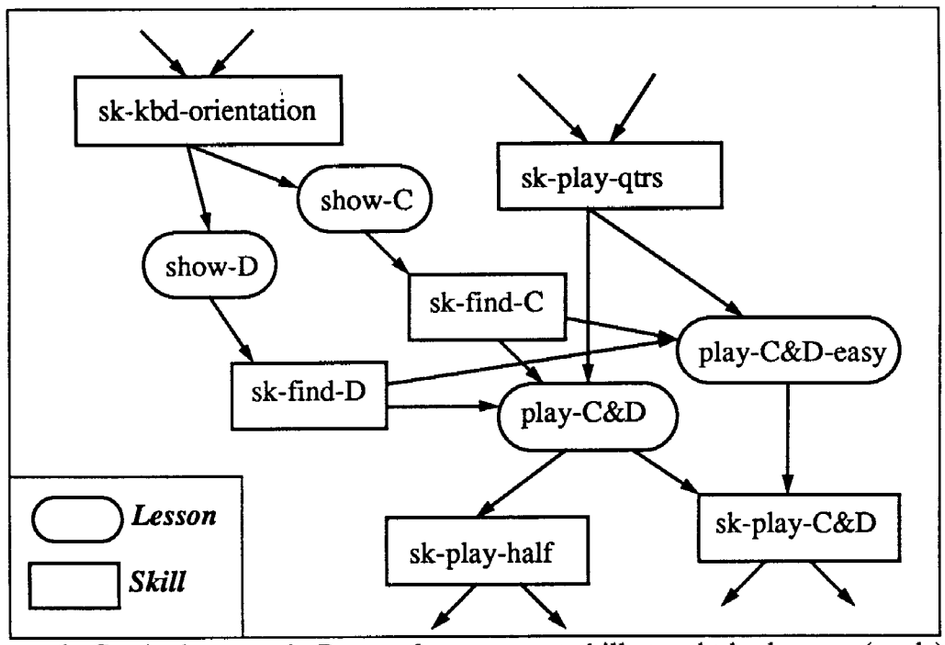

Figure 1: Curriculum graph. Rectangles represent skills taught by lessons (ovals). This partial curriculum is for a music tutoring system. Note that skills may be taught by more than one lesson, and a lesson may teach more than one skill.

We believe IDEAS technology offers a major new paradigm for education, not unlike the way that the spreadsheet has affected business analysis. Like the spreadsheet, IDEAS is highly domain- and genre-independent, so it can be applied to software, textbooks, and lectures in a variety of subjects. Also, like spreadsheets, IDEAS takes a traditional task and recasts it computationally, yielding an interactive design medium of great power.

We should note carefully what IDEAS is not. IDEAS is not simply a prototype system for course design. There are many tools on the market for building interactive educational software, and our system will in fact rely heavily upon commercial programming environments for constructing individual units of instruction. What IDEAS adds to existing systems are new advanced techniques for curriculum analysis and synthesis. IDEAS makes it possible to organize and coordinate instructional material generated by a variety of systems.

D. Relation to Ongoing Work

As noted above, IDEAS has it roots in Instructional Design theory, which among other things prescribes rules for educationally sound curriculum design. IDEAS makes these notions more formal and more concrete, enabling us to automate significant parts of curriculum analysis and synthesis.

IDEAS has at least some similarity to model tracing. Both approaches model student knowledge

as a set of skills, and use observable behavior to confirm that knowledge has been learned. Unlike model tracing, IDEAS does not require skills to be productions, so skills can represent arbitrarily complex chunks of knowledge. Because of this difference in granularity, IDEAS is useful for organizing large bodies of knowledge and searching for appropriate lessons. Also, IDEAS does not require extensive knowledge-engineering in the subject domain, and therefore, IDEAS-based systems should be easier to implement than model-tracing systems.

Like Shank's Case-Based Teaching Architecture, IDEAS can be viewed as a mechanism for indexing knowledge. Both architectures carefully limit their notions of knowledge and intelligence in order to avoid hard AI problems. In the case of IDEAS, the skill to be taught serves as an index into a curriculum that teaches, whereas in Case-Based Teaching, a problem situation serves as an index into stories about related experience. In both systems, an objective is to deliver a unit of instruction just at the moment the student is most prepared to receive it. IDEAS is fully compatible with the case-based reasoning (CBR) approach. For example, if CBR is used to create lessons, IDEAS can be used to ensure that these lessons cover the intended material, that lessons exist to prepare the necessary prerequisites, and so on.

IDEAS has interesting parallels to Conceptual Graph Analysis (CGA). Both can be viewed as techniques or methodologies for knowledge acquisition. IDEAS includes a formalized set of rules that can be checked by computer and which insure consistency and completeness in a curriculum design. Conceptual Graph Analysis is a less formal approach to eliciting and organizing knowledge in terms of taxonomic structures, spacial structures, and causal structures. CGA can be used to acquire a knowledge base, and IDEAS can be used to insure that the knowledge base is covered by a curriculum. We plan to investigate how CGA can be formalized and integrated with IDEAS. We suspect that the graph structures used in CGA may be ammenable to analysis similar to that used for IDEAS.

IDEAS is based in part on the Piano Tutor Project. This earlier work in multimedia and intelligent tutoring by Dannenberg and his colleagues forms the theoretical basis of IDEAS. In Piano Tutor the relationship between Instructional Design and Intelligent Tutoring was worked out and the precursor of IDEAS was built. For the Piano Tutor, we used a batch-mode analysis program to support curriculum design. While it did support the project, the program was intended for experts only, and we had no idea at the time that this work would extend to other domains.

The Piano Tutor teaches beginners to play the piano. It consists of an electronic piano with a computer interface, a computer and monitor, a videodisc player and second monitor, and an audio mixer to combine piano sounds, digitized voice from the computer, and sound from the videodisc. The Piano Tutor is by far the most sophisticated music-teaching computer system known to us.

The Piano Tutor assumes no initial musical knowledge, so the Piano Tutor must teach concepts such as notation, pitch, rhythm, and basic music structure in addition to piano performance skills. The basic approach of the Piano Tutor is to teach in units of short lessons that address one to a few concepts each. A lesson begins with a multimedia presentation to introduce the concept. Then, the Tutor asks the student to perform a task (usually a piece of music) that demonstrates mastery of the new concept. The Piano Tutor monitors the student's performance and offers suggestions if the student makes mistakes. When the new concept is mastered, the Piano Tutor selects a new lesson and the teaching process continues.

A pilot study used eighteen (18) students, staff, and faculty from the Carnegie Mellon

community. Ten (10) students completed the Piano Tutor curriculum, and eight (8) withdrew, mostly for reasons of personal schedules such as academic pressures. The logs support the hypothesis that the system could automatically tailor a curriculum to the needs of each student. Of the 10 students completing the curriculum, the time-to-completion ranged from 17 calendar days to 159 days, and from 3.7 on-line hours to 21 hours. These figures show that quick students or students with some background are not held back by a rigid or inappropriate curriculum. One might argue that the quick students simply play everything correctly on the first try and therefore take less time. However, the logs also show that the system gave more lessons to the slower students. For example, the number of presented music performance lessons ranges from 36 to 53. Clearly, the Piano Tutor is varying the lesson content according to the ability of the student.

No serious problems were encountered, and students were positive about the system. To a great extent, the numbers speak for themselves: most of the students stayed with the system to the end. Those that finished mastered what would normally be considered at least a one-year curriculum, and they could play piano at that level, based on informal testing away from the Piano Tutor. The *average* time was 69 calendar days and less than 20 on-line hours.

E. Project Objectives

VIII. The IDEAS Interactive Design and Analysis Tool

IDEAS will be packaged as an interactive design and analysis tool with the following components:

1. **A graphical/multimedia user-interface** that gives the designer the ability to enter, view, and edit a curriculum design. A curriculum design consists of skills (knowledge, concepts, and capabilities to be taught) and lessons (explanations, demonstrations, and activities that teach skills). Each lesson has a set of prerequisite skills and a set of objective skills. The interface will include links to authoring software for the implementation of lessons. A mock-up of the IDEAS curriculum design interface is shown in Figure 2.
2. **Curriculum analysis tools**, which are applied to the curriculum design. The relationships between skills and lessons are subject to formal analysis by computer. For example, the *useless skill* tool first finds lessons for which a given skill is a prerequisite. Then each skill that is neither a prerequisite to any lesson nor a final course objective is labeled as *useless*. A second example is circularity detection, which detects mutually dependent lessons. Other tools compute implication, equivalence, and other important relations among lessons and skills in the curriculum design.

IX. The IDEAS Lesson Selection Engine

IDEAS will also include a lesson selection engine for use in interactive, computer-based instruction. This software module selects lessons based on the student's history and skill level, the curriculum design, and the ultimate instructional goal. The lesson selection engine updates its model of the student on the basis of information reported at the end of each lesson.

A problem with automatic tutoring systems is how to place a student in the "middle" of a curriculum based on prior skills. We plan to explore and evaluate several solutions, including self-report and pre-test to establish a set of skills. When an advanced skill is asserted by self-report or pre-test, there will a set of implied skills. For example, if a student can play D-sharp on the piano, the student must also understand the concepts of sharps and flats. The curriculum analysis can

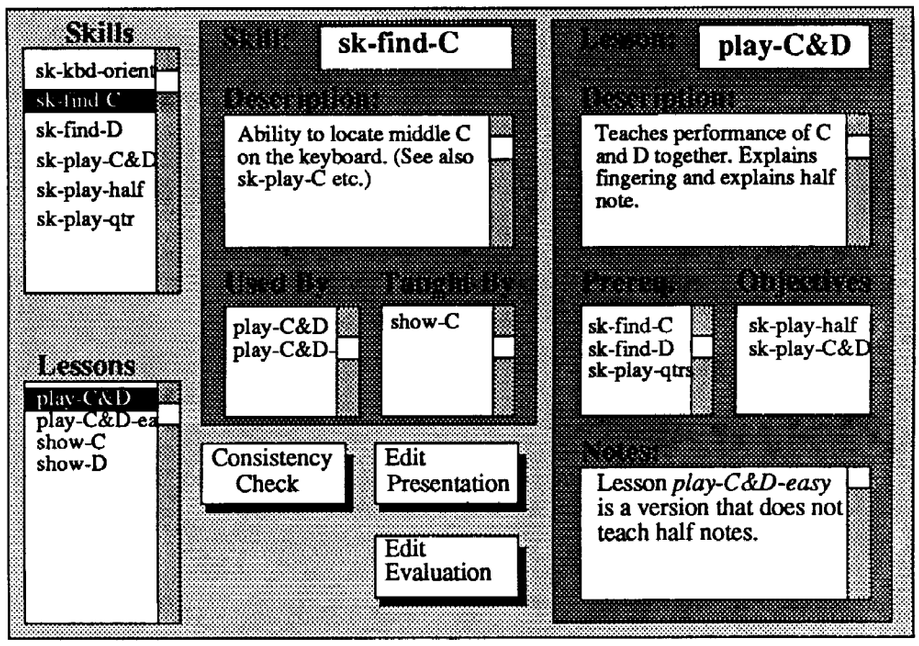

Figure 2: A mockup of the IDEAS curriculum design interface. Selecting skills and lessons (at right) brings up editable description panels (center and right). Not shown are output from global curriculum analysis and consistency checks.

determine these implied skills automatically. Curriculum analysis can also segment the curriculum into a set of levels and select important lessons from within each level. Presented in the right sequence, these lessons could form a pre-test for the curriculum. If the student starts at the wrong place in the curriculum, IDEAS can help the advanced student make rapid progress or the slower student "backtrack" to simpler material.

X. The Modern Math Course:

The modern math course will include:

1. A curriculum of freshman-level math appropriate for studying computer science,
2. A set of computerized materials that accompany a regular course offering,
3. Evaluation of the effectiveness of those materials,
4. Independent learning modules.

V. The Data Structures and Algorithms Course

The data structures course will include:

1. Adaptation of an existing data structures and algorithms course for self-paced and distance

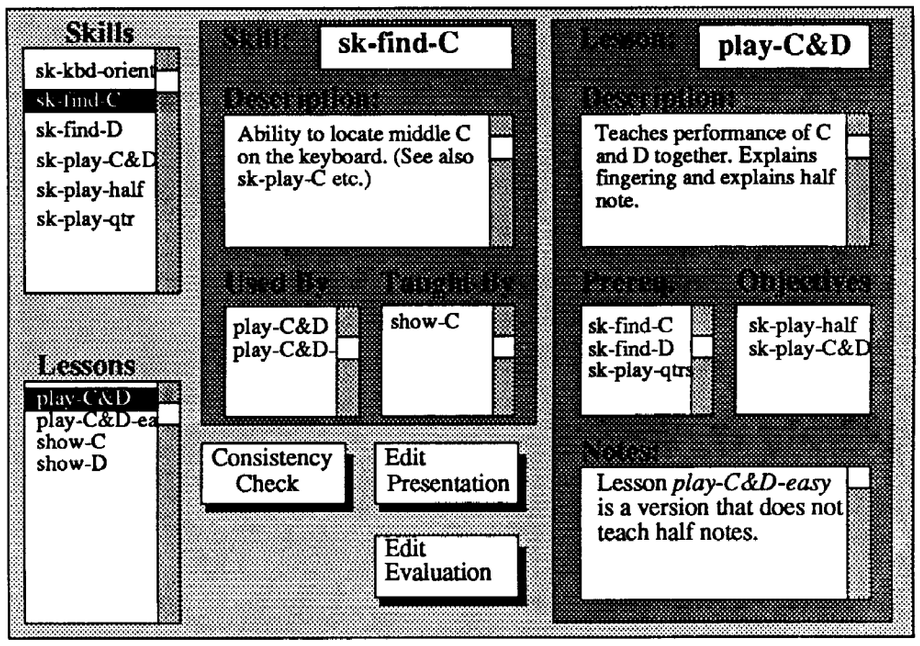

Figure 2: A mockup of the IDEAS curriculum design interface. Selecting skills and lessons (at right) brings up editable description panels (center and right). Not shown are output from global curriculum analysis and consistency checks.

determine these implied skills automatically. Curriculum analysis can also segment the curriculum into a set of levels and select important lessons from within each level. Presented in the right sequence, these lessons could form a pre-test for the curriculum. If the student starts at the wrong place in the curriculum, IDEAS can help the advanced student make rapid progress or the slower student "backtrack" to simpler material.

X. The Modern Math Course:

The modern math course will include:

1. A curriculum of freshman-level math appropriate for studying computer science,
2. A set of computerized materials that accompany a regular course offering,
3. Evaluation of the effectiveness of those materials,
4. Independent learning modules.

V. The Data Structures and Algorithms Course

The data structures course will include:

1. Adaptation of an existing data structures and algorithms course for self-paced and distance

delivery,

2. A set of computerized materials that accompany a regular course offering,
3. Evaluation of the effectiveness of those materials,
4. Independent learning modules.

VI. The Light Microscopy Training Project

The light microscopy training course will include:

1. Lessons in reagents, image acquisition and analysis,
2. A set of computerized materials and other distance learning materials,
3. Evaluation of the effectiveness of those materials.

F. Schedule and Milestones

I. Development of IDEAS - Years 1 and 2.

Underpinning the entire project is the software design environment. Since we have prior experience with the analsysis algorithms, we anticipate a working version in the first year. IDEAS will be evaluated for ease of use by curriculum builders. Evaluation includes, but will not necessarily be limited to, the work of design teams for the Modern Math Curriculum, the Data Structures and Algorithms course, and training in advanced Light Microscopy. The lesson selection engine is less critical early on and will be implemented in year 2.

II. Development of the Modern Math Curriculum. Years 1 - 3.

A course in modern math that is supported by Mathematica is a priority for the university and is underway, supported by internal funds. Students will be engaged in constructionist symbolic computation. In the first year of the project we anticipate subjecting the Modern Math Curriculum to formal analysis by IDEAS. This experience will give us insight into both IDEAS and the Modern Math Curriculum. The former will be analyzed for usability and utility. The latter will be analyzed for consistency and completeness. In the first year of the project, Modern Math will be taught as a traditional classroom course, augmented by a computing laboratory. Based on formal analysis and experience, independent learning modules will be lifted from this curriculum during the second year. Distance learning materials will be comprised of successful modules.

III. Use of IDEAS in Data Structures and Algorithms Years 1 - 3.

The course in data structures and algorithms will be a much more standard course than the Modern Math offering. One major difference is that data structures and algorithms will be delivered by an intelligent multimedia tutor. We plan to design the course in year 1, implement the interactive multimedia in year 2, and present and refine the course in year 3. The entire course is suited to independent study and distance learning.

IV. Use of IDEAS in Light Microscopy Training Years 1 - 3.

The first objective is to develop curricula that teach the use of the multimode light microscope and the standing-wave light microscope. Both are state-of-the-art instruments that permit posing and answering previously intractable scientific and clinical questions. Timely diffusion of these important innovations depends, in large part, on the quality of training curriculum designs. Standard practice is to have on site training seminars, informal training, and selected off site

colloquia. The inefficiencies of this practice are obvious and the use of IDEAS, especially coupled with distribution via the National Information Infrastructure could help revolutionize training in the biomedical community.

V. Evaluation of the Effectiveness of IDEAS Years 2 - 3.

Evaluation is key to the project. Evaluations will be performed to understand the capabilities of IDEAS both as a course synthesis tool and as a course analysis tool. To understand the role of IDEAS in course synthesis, we will collect usability data as it is applied in designing curricula for the three content areas. IDEAS is intended to provide principled support of the design process and we will document, with these case studies, the ways in which the curriculum designers find it makes the process more efficient and less error prone. In particular, we will survey designers as to what particular features they find most and least helpful.

The participants in the design of the Piano Tutor curriculum reported that the design tools at that time (which are a specific subset of what IDEAS is to be) were very helpful in catching and repairing errors. The piano teaching experts originally planned their curriculum on paper and when this curriculum was translated onto the computer, the design and analysis tools helped to identify numerous omissions and errors in curriculum and also facilitated their repair. We want to better document these error correction and design facilitation events to add to this anecdotal evidence more solid evidence of the effectiveness of IDEAS.

To understand the role of IDEAS as a course analysis tool, we will perform experiments to see how well new users can use IDEAS to analyze curricula with known errors. In addition, we will apply IDEAS to existing curricula, for example, data structure and algorithm textbooks. We believe we will find numerous errors in such curricula and thus provide further evidence for how IDEAS can improve upon the non-computer-aided design process.

G. Budget Justification

a. Personnel

Roger B. Dannenberg received a Ph.D. in Computer Science from Carnegie Mellon in 1982, and he is now a Senior Research Computer Scientist in the CMU School of Computer Science. He was a Principal Investigator and Technical Director of the Piano Tutor Project. Dr. Dannenberg also participated in the design of Garnet, a system supporting advanced graphical user interfaces. His work included extending the prototype/instance model to structures of objects (such as complex graphical displays), the design of the Lapidary interface builder, the concept of constraint specification by example, and debugging and inspection techniques for constraints. Dr. Dannenberg is also the co-author of *Multimedia Interface Design*, reportedly one of the best sellers of the ACM Press.

Dana Scott is Hillman University Professor of Computer Science, Philosophy and Mathematical Logic at Carnegie Mellon University. He is a faculty member in the School of Computer Science and in the Departments of Mathematics and Philosophy. He received his B.A. in Mathematics from the University of California, Berkeley, and his Ph.D. in Mathematics from Princeton University. Before joining the faculty at Carnegie Mellon, he was Professor of Mathematical Logic at Oxford University, and previously held appointments at Chicago, Berkeley, Stanford, Amsterdam, and Princeton. Dr. Scott was awarded an honorary doctorate from the University of Utrecht in the Netherlands in Philosophy.

Professor Scott is a Fellow of the British Academy, the U.S. National Academy of Sciences, the American Academy of Arts and Sciences and the American Association for the Advancement of Science. His memberships include the New York Academy of Sciences and Finnish Academy of Sciences and Letters. He received the LeRoy P. Steele Prize of the American Mathematical Society and was recipient (jointly with M. Rabin) of the ACM Turing Award. He has been a Bell Telephone Fellow (Princeton), a Miller Institute Fellow (Berkeley), a Sloan Research Fellow, a Guggenheim Foundation Fellow, and was granted a Senior U.S. Scientist Award by the Humboldt Foundation in Germany. He is on the editorial boards of several technical journals and is a series editor for Springer-Verlag and Oxford University Press.

His work in logic has concerned mainly model theory, automata theory, set theory, modal and intuitionistic logic, constructive mathematics, and connections between category theory and logic. His interests in philosophy and linguistics concern the foundations of logic, the philosophy of mathematics, the semantical analysis of natural language, and applications of computers to natural-language processing. His work in computer science has been directed toward the development of denotational semantics of programming languages and the mathematical foundations of a suitable theory of computability. He is also interested in more practical questions of program specification, verification, and development and in symbolic computation. He has supervised (sometimes jointly with other colleagues) thirty-five Ph.D. theses within this range of subjects. He has recently become particularly interested in new developments in teaching with the support of symbolic computation. In this capacity Professor Scott served this past year as Director of the Research Institute for Symbolic Computation at the Johannes Kepler University in Linz, Austria.

Kenneth R. Koedinger has a M.S. in Computer Science and a Ph.D. in Psychology and is now a Research Computer Scientist at Carnegie Mellon. He has been developing and evaluating "cognitive tutors" for high school mathematics in collaboration with John R. Anderson. Cognitive

tutors are a type of intelligent tutoring system where the design of the system and its computational core are based on a cognitive model of student users. Dr. Koedinger developed a cognitive tutor for geometry theorem proving, called ANGLE, based on empirical research and a cognitive model of the visual planning strategies of expert theorem provers. ANGLE has been subject to formative evaluations in the laboratory and a summative evaluation at Langley High School in Pittsburgh. In the latter, students using the tutor scored one standard deviation better than students in normal classrooms (approaching the effectiveness of individual human tutors). ANGLE also led to positive motivational and social consequences in this urban high school, including motivating students to voluntarily come to the computer lab during lunch and after school to use the system. ANGLE continues to be used in geometry classes at Langley.

Dr. Koedinger is also participating in the development and evaluation of a cognitive tutor for Algebra that is currently being used by 12 classes and about 250 students at Langley. In addition to tutoring students to formulate real world problem solving situations as algebra equations, the tutor helps students learn to use modern computational tools (a spreadsheet, grapher and equation solver) to help solve problems. To dispel the concern that cognitive tutors are limited to instruction symbol manipulation skills, Dr. Koedinger's newest project is to create and evaluate a cognitive tutor for mathematical investigation and discovery. This system uses interprocess communication to tutor students in the use of the Geometer's Sketchpad, an existing software tool for geometric visualization and exploration. Based on the successes of these systems, the Pittsburgh Public School district recently decided to purchase 50 more Macintosh workstations to create computer labs are 2 more high schools.

Philip L. Miller has been Director of Introductory Programming at Carnegie Mellon since 1979. He is active in the ACM's SIGCSE. He was charter member and later Chairman of the College Board's Advanced Placement Computer Science Development Committee. Dr. Miller has been Principal Investigator of the GNOME, MacGnome, and ACSE software development projects. Dr. Miller was Director of Carnegie Mellon's Center for Art and Technology and is currently on the Executive Committee of Carnegie Mellon's Center for Light Microscope Imaging and Biotechnology - an NSF Science & Technology Center.

D. Lansing Taylor was educated at the University of Maryland (B.S., 1968 - Zoology) the State University of New York at Albany (Ph.D., 1973 - Cell Biology), and Woods Hole Marine Biological Laboratory (Postdoctoral, 1973 - Biophysics). He is currently Director, "Center for Light Microscope Imaging and Biotechnology", NSF Science & Technology Center (1991-present). Other professional experience includes Woods Hole Marine Biological Laboratory, Research Associate, 1973-1974. Harvard University, The Biological Laboratories, Assistant Professor, 1974-1978. Stanford University, Department of Structural Biology, Sabbatical Leave, Spring 1980 - Dr. Lubert Stryer and Dr. Jim Spudich. Harvard University, The Biological Laboratories (a) Associate Professor, 1978-1982. (b) Chairman, Graduate Committee, 1980-1982. Instructor, Marine Biological Laboratory (Woods Hole) (a) Quantitative Optical Microscopy Course, 1980-1990. (b) Physiology Course, 1984. Carnegie-Mellon University (a) Professor of Biological Sciences, 1982-present. (b) Director, Center for Fluorescence Research in Biomedical Sciences, 1981-1991.

Professor Taylor has extensive experience as editor of scholarly publications. Professional Service Editor, Cell Calcium, 1993 -. Editor, J. Cell Biol., 1981-1985. Editor, Cell Motility, 1980-. Editor, NeuroImage, 1991-. Editor, J. of Fluorescence, 1991-. Editor, Seminars in Cell Biology,

1989-. Editor, Vol. 29 & 30, Methods in Cell Biology, 1988. Editor, Current Topics in Cell Biology, 1990. Corporation Member, Marine Biological Laboratory, 1974-. NIH Molecular Cytology Study Section, 1985-1989. Consultant for Light Optical Methods - NIH Grants, NSF Grants - 1975-. American Society for Cell Biology, Publications Committee, 1988. NSF Panel - Biological Research Center Grants, 1988. NSF, NIH Grant Reviews for Instrumentation Scientific Advisor, West Penn Hospital and Foundation Basic Science Advisory Committee, Pittsburgh Cancer Institute Council Member, Biophysical Society External Advisory Committee, Cytometry Group, Los Alamos National Labs Professional Societies New York Academy of Sciences American Society of Cell Biology Biophysical Society Phi Sigma Society American Association for the Advancement of Science International Society for Analytical Cytology

He has patented Standing Wave Luminescence Microscopy. Pat. #. 4,621,911, with F. Lanni, A. Waggoner.

Ten Most Recent Publications (118 total) Conrad, P., K. Giuliano, G. Fisher, K. Collins, P. Matsudaira, and D. L. Taylor,1992. Relative Distribution of Actin, Myosin I, and Myosin II during the Wound Healing Response of FibroblastsJ. Cell Biology 120: 1381-1391

Kolega, J., M. Nederlof and D. L. Taylor, 1993. Quantitation of Cytoskeletal Fibers in Fluorescence Images: Stress Fiber disassembly accompanies dephosphorylation of the regulatory Light Chains of Myosin II. BioImaging (in press)

Gough, A. and D. L. Taylor, 1993. Fluorescent anistropy imaging microscopy maps calmodulin binding during cellular contraction and locomotion. Journal of Cell Biology. 121:1095-1107

Hahn, K., P. Conrad, J, Chao, D. L. Taylor and A. Waggoner, 1993. A photocross-linking fluorescence binding during cellular contraction and locomotion. Journal of Histochem.Cytochem. 41

Kolega, J., K. and D. L. Taylor, 1993. Gradients in the Concentration and Assembly of Myosin II in Living Fibroblasts during Locomotion and Fiber Transport. Mol. Biol. of the Cell

Janson, L. W. and D. L. Taylor, 1993. In Vitro Models of Tail Contraction and Cytoplasmic Streaming in Amoeboid Cells. J. of Cell Biol. (in press)

Betzig, E. R. J. Chichester, F. Lanni and D. L. Taylor, 1993. Near-Field Fluorescence Imaging of Cytoskeletal Actin. BioImaging (in press)

Lanni, F., B. Bailey, D. Farkas and D. L. Taylor, 1993. Excitation Field Synthesis as a Means for Obtaining Enhanced Axial Resolution in Fluorescence Microscopes. BioImaging (in press)

Giuliano, K. A. and D. L. Taylor, 1993. The Interaction of Actin with Profilin Optimizes the Formation of a Gradient of Actin Assembly in Cells. J. Cell Biol. (submitted)

Bailey, B., D. L. Farkas, D. L. Taylor and F. Lanni, 1993. Enhancement of Axial Resolution in Fluorescence Microscopy by Standing Wave Excitation. Nature (in press).

b. Equipment and Facilities

Modest equipment requests are made. Facilities charges are computed as a function of personnel and time.