Project Athena at MIT: An Overview

by Sherry Turkle and Donald Schön

May 1988

Introduction

In 1983, MIT began a large scale experiment in the uses of the computer for undergraduate education. The goals of Project Athena were ambitious: technical, educational, and even epistemological. The first priority was to create a "coherent network" for educational computing, consistent in its "courseware," hardware, operating system, and programming languages. The goal was to have a student's one time investment in learning the system provide access to the full range of educational software developed at MIT. Building on this technically sophisticated network, Athena's architects hoped to integrate "modern computer and computational facilities into all phases of the educational process" in order to "help students learn more creatively and fully in a wide range of disciplines" by developing "new conceptual and intuitive understanding and [improving and refining] our teaching methods."[1]

In Spring 1986, the Project Athena Study Group, a faculty committee chaired by Professor Jean de Monchaux, mandated a study of the impact of Athena on teaching and learning in four MIT departments: Architecture and Planning, Chemistry, Civil Engineering, and Physics by principal investigators Professor Sherry Turkle and Professor Donald Schön and staff members Brenda Nielsen, M. Stella Orsini, and Wim Overmeer. These studies began in April 1986 and were completed in Fall 1987.

The goal of the study was to understand the educational impact of Project Athena, including students' experience of the new educational software and the process of introducing computers into learning environments. Thus, our interviews with faculty and students cast a broad net. We discussed personal experience with computers as compared with other tools, concepts of computer literacy, perceived advantages and disadvantages of computing, and how faculty and students saw the implications of the technology for education, research, and practice. In our interviews we combined guideline questions and an open ended format. This allowed us to collect comparable data from each respondent while permitting flexible discussion of topics of special concern to particular individuals.[2]

[1] *Project Athena, Faculty/Student Projects, MIT Bulletin,* March 1985.

[2] Students were promised anonymity, and their names have been fictionalized or suppressed. Faculty were told that when specific attributions were made, they would have an opportunity to review our written account of their statements.

The writing of our final report coincides with the end of Project Athena's first five years. It is a good time to review what we have learned. Our purpose in this overview is to present a set of organizing thoughts that will help in thinking through principles for future action. We present findings from our four case studies. This overview is not a summary of but a reflection on the cases.

Athena has greatly increased MIT undergraduates' access to computers and stimulated the development of a number of exciting projects. At the same time, Athena has made some policy decisions that have proven problematic. They rest on habits of mind that, if carried further, could prevent Athena from realizing its full potential.

Part I. Learning From What Has Gone Right

Athena has had significant achievements. We found projects to which students and faculty have responded positively—in ways that sometimes conform to the developer's intentions and more often, go beyond them. Looking at these projects offers intimations of success—potentially rewarding directions for the further development of Athena's educational experiment.

Here we briefly touch on a few exciting projects—GROWLTIGER, a program for the analysis and design of indeterminate structures developed in Civil Engineering; the computerized infrared spectroscopy experiment in Chemistry; the development of computer based design systems in the Computer Resources Laboratory in the School of Architecture and Planning; and the use of the computer in the Junior Laboratory in Physics.

The consideration of examples such as these leads us to an analysis of what has worked, why it has worked, and how we can build for the future. In such projects there is no doubt that the computer functions as a tool. It does things that people could otherwise do only with great difficulty, and does them much faster. But what is most interesting about these projects is that the computer not only helps to do the old things more efficiently, but provokes new ways of seeing. It is "more than just a tool." The computer is evocative: you may be led to reflect on your thinking, restructure your view of professional practice, and take a new look at the knowledge behind the practice.

More Than Just a Tool

It has long been commonplace to use computers to collect laboratory data in order to save time, keep students from getting lost in the mechanics of calculation, and help them remember what the experiment is really about. These goals were certainly achieved in the infrared spectroscopy experiment in Chemistry and the Junior Laboratory in Physics where the time required to perform experiments was reduced by a factor of five or six. But in both of these cases the efficiency of the computer for taking, processing, and analyzing data also gave rise to a different kind of benefit.

In the infrared experiment, for example, the computer is used to graph the infrared spectrum of acetylene gas. The PEAKFINDER program then analyzes the data, scans the curve, and assigns peaks. Further, it assigns numerical values to the peaks and calculates the relative distance between them. Students can then match actual values with the values predicted by quantum mechanics. Doing these things by hand used to take many hours, even days. When they use the computer, students have the time to think about the meaning of their data. In addition, because the program gives them more detailed information and a more precise visual display of the peaks, it allows them to make new connections between theory and data. Students have the feeling that they are confirming the theory for themselves. As a result of such experiences, many students said that the theory became "real" to them for the first time.

The computerization of the Junior Laboratory in Physics presents a variation on these themes. Here, both students and instructor emphasized a paradoxical result: the use of the computer to collect and process data gave students a better feeling for the "real world" of physical phenomena. The computer allowed students to run through many variations of the same experiment. The fact that they could make quick calculations allowed them to implement thought experiments which gave more of a feeling for what was going on than they had ever had before.

With the computer, students are also better equipped to handle uneven and anomalous data. Before the computer entered the laboratory, if a student's one run of an experiment yielded only anomalous data, the student could not bring the result of experiment into relationship with theory. To make this connection, the student would be forced to rely on packaged data. Physics faculty stress that the "real world" is not neat and the scientist's primary allegiance is to messy data. This was always a problem for students who were meeting theory for the first time and needed "clean data" in order to see the connection to theory.

In both the infrared spectroscopy experiment and the Junior Laboratory in Physics, the computer, by being a magnificent tool, becomes more than just a tool.

New Ways of Thinking

Designing is a kind of thinking you do with a tool. Traditionally, designers have done their thinking with the aid of a pencil or with scale models. It comes as no surprise to designers that their tools have a profound effect on the way they go about their work.

In the Computer Resources Laboratory in the Department of Architecture and Planning, a small family of computerized design assistants has replaced or complemented the pencil in studios devoted to architectural design and site planning. We have found that these new tools influence not only how designers go about their work but how they think through problems.

Different faculty and students describe this influence in somewhat different ways. One faculty member highlighted the fact that the computerized system requires students to make design decisions explicit: "It forces them to think very clearly about modes of transport and about relationships and priorities of locations ...; [it encourages them] to conceptualize a city in ways that can be embedded in a computer system." Another faculty member noted that the computerized design assistant can help students get unstuck when they reach an impasse because it allows them to make quick changes "on the order of magnitude of turning your paper upside down." A third described how the "drawings" students make on the computer by aggregating simple shapes into larger modules can lead students to produce novel site plans; the computer "liberates" students' visual imagination. For another, working in electronic environments where "information is stored in many forms and in many locations" means that students are forced to develop rich cognitive models of computational environments which then enhance their general learning ability. And for a professor of Planning who uses spreadsheets to teach modeling, computers make it possible to "open the black box" of a model and develop a deeper understanding of its internal dynamics. Faculty and students within Architecture and Planning have reservations about the computer presence. But despite their doubts they tend to give the computer credit for encouraging creativity and providing deeper insight into design and modeling.

Microworlds

The computer provides an opportunity to build self-contained environments—simplified representations of some real or imagined world. When you are in such a virtual world, you can create and manipulate objects that are subject to its laws. The educational potential of such environments, or microworlds, has long been evident.

In our studies, GROWLTIGER was one of the most interesting examples of a microworld. With it, a Civil Engineering student can build a visual representation of a structure such as a truss or a frame, determine its geometry and dimensions, and select the loads to be applied. A finite element program analyzes the effect of the loading and produces a visual display of the resulting deflections.

GROWLTIGER was easy for students to learn and use. And, as in the infrared spectroscopy experiment, students found it a useful tool. It saved them hours they would otherwise have spent in cumbersome and error-prone calculation. But the tool had meaning beyond this. First, it was seen as a device for designing in a department that left many students thirsty for design experience. Second, GROWLTIGER enabled students to make many small changes in proposed structures and quickly see their effects. Students spoke of "getting a feel" for the behavior of the structure and of coming to understand structural theory in a new way. Not infrequently, experimenting with GROWLTIGER confronted students with surprises—for example, discovering that removing components of a structure can actually increase its stiffness. Such surprises provoked reflection, leading to new experiments and new insights into the implications of structural theory for design.

Students' positive responses to GROWLTIGER can be contrasted with their reactions to MACAVITY, an "intelligent tutor" designed to detect and correct the causal reasoning that students employ in the analysis of determinate beams. Students found MACAVITY boring. They complained that MACAVITY didn't allow them to do anything creative, so they turned their creativity against it: by reprogramming it to solve problems for them, they converted it from a tutor to a slave.

For MIT students, it would seem that microworld strategies can be educationally effective because they don't pretend to teach students but allow students to operate actively and freely within a computer environment. Students find them compelling—"they have holding power." Microworlds allow students to "play around," to manipulate variables, to be surprised. They provide a concrete "object to think with."

Personal Appropriation and Styles of Mastery

Even among educated and technically sophisticated people, there is a widespread notion that the computer can be effectively used in only one way. Although there is much talk about the computer's versatility and flexibility, this rhetoric usually coexists with a conflicting assumption that by its nature the computer imposes a particular way of working: rule-driven, formal, and proceeding from global to local.

In our studies we have found that people use the flexibility of the computer environment to work in very different ways. Diversity rather than homogeneity characterizes students' and faculty's ways of using the same machine and software. In the case of students, diverse styles are evident not only in what they do with computers, but in their feelings about what they do. Diversity shows up in students' different computational aesthetics: what they find fascinating, valuable, exciting, frustrating, and useless about the computer. For example, in contrast to the stereotypical style of computer use which we capture by talking of "planners," some students approach the computer as though they had just entered a foreign territory that exists for the pleasure of its exploration. They work without a precise plan, often beginning with one element and playing with it, allowing one idea to lead them to the next. They let their first idea change into something else or into nothing at all. These are more bricoleurs, or tinkerers, than planners.[3] For example, one graduate student in Architecture approaches design as an exercise "where I randomly ... digitize, move, copy, erase the elements—columns, walls, and levels—without thinking of it as a building, but rather as a sculpture ..., and then take a fragment of it and work on it in more detail."

Thus, in our studies we have seen the computer function both as an evocative object that provokes new ways of thinking and as a projective object that reflects the personality and cognitive style of the user. If given an opportunity, different people make the computer their own in their own ways. The successful projects we have named above are characterized by the ability they give their users to personalize an involvement with the machine. It is perhaps most dramatic in design;

[3] For a more complete description of styles of mastery and the phenomenon of the computer as a projective screen or "Rorschach," see Sherry Turkle, *The Second Self: Computers and the Human Spirit* (New York: Simon and Schuster, 1984), especially Chapter 3; "Computer as Rorschach," *Society*, January-February 1980; and "The Subjective Computer," *Social Studies of Science*, 12, 2, May 1982. The theme of diversity in styles of mastery runs through our case studies where we simplify the wide range in styles and limit ourselves to a discussion of what Turkle describes as the opposition between bricoleurs and planners.

but in all of the most successful projects, students found some way to personalize the computer. And they did so even, as in the case of GROWLTIGER, when the computer environment was highly constrained.

* * *

The designers of Athena believed that they could multiply the computing resources available for undergraduate education and turn the talent and energy of the MIT faculty toward educational computing in a straightforward way. Part of what made it seem so straightforward was the assumption that at MIT this kind of effort would be welcome. But the situation was in fact more complex. To take only one issue among many, but one which came up with considerable force in our case studies, even at MIT computers evoke significant ambivalence among the faculty. In each of the disciplines we studied, we found a "sacred space"—a core of intellectual work, artistry, or creativity—that faculty tried to protect from the "intrusion" of the computer. The definition of sacred space varied from field to field, but no field was without it. In Architecture, many faculty resented the computer's intrusion into the artistry of design. In Chemistry, most faculty actively resisted the idea that computers could usefully enter the lecture hall. The development and presentation of theory were to be kept inviolate—computers were to be kept in their place, and that was in the lab. In Physics, some faculty were hostile to the idea that the machines would bring computational ways of thinking into reasoning about physical phenomena. Others feared computer simulations would mask the "real world." In Civil Engineering, some faculty deplored the use of computer software for structural analysis because it might blind engineers to crucially important sources of error and uncertainty. In all these cases, faculty saw no contradiction between their wish to protect a "sacred space" from the computer and their routine acceptance of the computer as an indispensable research tool.

In addition, although MIT is marked by the energy, dedication, and exuberance of a "hacker culture" and has immensely profited from that culture's historical devotion to computation, there is also considerable ambivalence about hackers and what they have come to represent. Here we characterize the hacker culture as one that has a love of the computer "for itself," not only for what it can do, but for its purely artistic and expressive value. The cost of this passion is that it is easy to become lost in the technical. MIT faculty and students see hackers all around them (including on the Athena staff) and many define themselves in opposition. Across the disciplines we studied, faculty and students have a way of

talking that tends to define the serious disciplinary scholar as a "not-hacker."

For example, in Architecture, some describe the tension between "being a good programmer and being a good architect" as though the technical sophistication of the one would impinge on the artistry of the other. Others fear that if they become too competent, the balance of their practice would shift to programming: "I don't want to become so good at it that I'm stuck in front of a computer 40 hours a week. It's a matter of selective ignorance."

Given such complex motivations for reticence, introducing educational computation on a wide scale at MIT was going to be difficult. But Athena increased its difficulties through a series of administrative decisions. If people are primed to see the computer as a threat to something they see as precious and vulnerable in their discipline, then Athena policies that deprive them of control over how the computer is to be used are bound to increase the level of perceived threat. And in its efforts to create a coherent and centralized system, Athena often worked against the grain of locally developing computer cultures. This, too, was bound to make people more fearful. Athena adopted institutional strategies that increased rather than decreased resistance to its goals, leading some faculty not to join up and causing other faculty to be marginalized and isolated from their departments when they did participate. These institutional strategies have been roadblocks; they reflect habits of mind and patterns of approaching problems that run counter to what works best at MIT.

Part II. Roadblocks

Centralization

Although its intention was to "let a thousand flowers bloom" by encouraging a diverse group of faculty experiments, Athena paradoxically chose to do so in a centralized way.

* Organizational Centralization

Athena created a central institutional mechanism to elicit, screen, and fund faculty proposals. It circumvented the departments, causing widespread resentment among faculty members. Athena tried to draw directly on the enormous energy, talent, and entrepreneurial strengths of the MIT faculty. But Athena's centralized organization, widely perceived as an alien layer of bureaucracy, undermined its ability to do so.

* Physical Centralization

The physical setup of Athena led some potentially interested faculty members to stay away. In the interest of efficiency, Athena established centrally located computer terminal rooms which had the effect, at least in part, of keeping work stations from faculty offices, classrooms and laboratories, thereby distancing faculty members from their own educational initiatives.

* Technical Centralization

Since one of its original goals was to build a coherent computer network, Athena placed restrictions on the hardware, operating systems, and programming languages that could be used in its funded projects. The decision, for example, to make UNIX the only Athena operating system reflected an underestimation of the strength and persistence of the DOS-based personal computer culture at MIT. Many faculty were experienced and quite comfortable with DOS which they used for research and writing. DOS was, in many cases, the system they used in their first "private" efforts to integrate computers into their teaching. Many faculty came to see the DOS/UNIX controversy as symbolic of Athena's attempt to control their use of computers. Some faculty avoided Athena because of UNIX—the cost of learning a new operating system was too great; others avoided Athena because it would not support BASIC or other languages they were used to; still others resented the fact that they could not use hardware from manufacturers other than IBM and DEC. In particular, many wanted to use Apple machines such as the Macintosh.[4]

The cultures of departments, and in some cases, subdepartments, are very different. This makes their computer cultures very different too. Athena was not set up to exploit the creative potential of this diversity. It was set up to impose its own rules for how things should be done. In fact, a coherent computing network has not yet emerged and seems unlikely to do so, at least in the near future. IBM ATs and RTs, DEC work stations, specialized CAD/CAM machines and Apple Macintosh computers are all in operation in the Computer Resources Laboratory. Three different operating systems are in use in Civil Engineering. In part, this reflects factors beyond Athena's control. Both DEC and IBM were late

[4] In our interviews with faculty, we found that many saw this as Project Athena policy, especially in the early years, interpreting in this light decisions on the part of one or more of Athena's proposal review committees and the rationale given for not funding proposals. Athena's Executive Committee never officially adopted this policy which emerged, nevertheless, as a policy-in-use. Understanding how this could have happened should inform Athena's next phase of development.

in delivering the work stations they had promised; faculty involved in Athena-supported projects had to use available hardware and adapt it to their purposes as best they could. But more fundamentally, the failure to achieve a coherent educational computing network reflects the existence of diverse computer cultures at MIT—a diversity that Project Athena tried unsuccessfullly to fight.

Artificial Boundaries

In order to promote undergraduates' access to computers, Athena tried to funnel its resources toward teaching rather than research, and toward undergraduate rather than graduate education. But in many instances, faculty's educational initiatives in undergraduate education are inseparable from their larger enterprise of teaching and research. Athena's attempts to enforce its conception of appropriate boundaries led, in many instances, either to the exclusion of faculty or to artificial distinctions that were not followed through in practice.

Technocentrism

In spite of Athena's intention to emphasize educational priorities, in practice, its emphasis on technically sophisticated workstations and software tended to subordinate educational considerations to technical ones. Its high-tech focus, so characteristic of the MIT ethos, did not make Athena a welcoming environment for the "mundane" uses of the computer (such as word processing and electronic mail) that made up the bulk of student demand. Athena gave priority to projects that could exploit the computing power of its work stations. This fostered a heightened distinction between technically sophisticated and technically naive uses of computation which discouraged faculty who wished to make what they saw as educationally interesting uses of routine technology. They felt like second class citizens. Faculty and students outside the School of Engineering tended to see the "Athena style" as an "engineering style" and resented its imposition. Athena conveyed to many faculty members the message that, as one of them put it, "Athena is not about education; it is about technology and logistics."

The technical concerns of Athena's central office left the specifically educational dimension of their projects to faculty. But faculty, too, have been largely preoccupied with technical problems. They have struggled to get systems working, to debug hardware, operating systems, and software. And they have struggled to keep up with technical rules of the game that kept changing as new features came on line. Many faculty ended up feeling that they were getting the

worst of both worlds: Athena was not set up to support their educational concerns. Nor could it be successful at providing an efficient and reliable computer facility because it was in the process of developing a new system.

Educational Innovation: An Exercise for the Left Hand

Implicit in Athena's strategy was an assumption that MIT faculty would be able and willing to devise new educational software in addition to doing their usual jobs as something of "an exercise for the left hand." There was little if any recognition that a faculty member's decision to take on an Athena project might represent an important shift in career direction. Indeed, Athena was widely perceived—especially in the School of Science and in the Project's early years—as refusing to fund salaries that would free faculty from teaching responsibilities. But in the perception of most faculty members who tried, development of new educational uses of the computer was a demanding activity. It took intellectual and administrative effort. And it took a great deal of time. Feeling that they had to get in all the way or not at all, many of them chose to opt out.

Although Athena assumed that faculty could do experiments in educational computing "on the side," with few exceptions we found that the more interesting projects followed the opposite pattern. They were outgrowths of the major research interests of senior faculty who had been long-time champions of educational computing. Sometimes these faculty worked in tandem with junior colleagues. But the latter, tempted to follow in the paths of their senior mentors, have had to reckon with departmental incentive systems that strongly discourage such a move.

It is almost impossible to remain a research leader within one's field and, at the *same time*, to devote the requisite effort to developing a serious innovation in educational computing. To succeed, one has to make educational computing one's "business," at least for a while. But faculty members who try to do so get caught in a trap.

Typically and quite naturally, they do not know at the outset how to do research in educational computing. But then, as they learn, they are seen by their colleagues as diverging from the path to success within their departments. Furthermore, even if their new educational research produces results, who in their departments would be equipped to evaluate them? MIT has yet to face up to the problem of creating institutional conditions that would support faculty in turning their "right hands" to the achievement of Athena's educational objectives.

In the light of our findings, it is understandable that the developers of Athena projects tended to be either individuals marginal to their departments, or "short timers" who had been asked to devote effort to Athena but had no intention of following up the possibilities created by their work, or long-time champions of computers in education.

Part III. Where Do We Go From Here?

We began by identifying strengths that Project Athena can build on, and we have also described institutional patterns that serve as roadblocks to realizing the full potential Athena has created. The case studies that follow suggest several directions for future action that we believe would increase the likelihood of success in the next phase of the Athena experiment.

1. The boundaries between Project Athena and other kinds of faculty involvement with computers should be made more permeable.

Athena should open up to include a far wider range of faculty initiatives in the development of educational software—regardless of the type of hardware, operating system, or programming language employed.

Athena should recognize that computers are part of the total intellectual life of the faculty member. In many of the most interesting Athena projects we observed, graduate and undergraduate teaching are intimately associated, as are research and teaching. Athena should not try to confine itself to one component of the educational enterprise; artificial boundaries of this sort are destined to become permeable in practice, whatever the formal policies may be. Faculty members need to have the computer where they are, both intellectually and physically. So, for example, if this means that faculty members like to work with students in their offices, this is where the computers must be.

2. Departments are best equipped to choose the hardware, operating systems, and programming languages applicable to their fields. Locally managed "natural selection" should take precedence over centralized planning. Athena should encourage departments to create their own structures to stimulate and guide the development of educational uses of computers, as some departments have already done informally. At the same time, faculty members operating in diverse fields should be encouraged to learn from each other through seminars, workshops, and other vehicles for information dissemination.

3. Indeed, Athena should be alert to situations where the educational content of its projects suggests collaboration across departments. It is already apparent

that certain broad themes of common interest are shared by Athena projects in different departments. Two of the more interesting of these are uses of the computer for the collection, processing, and analysis of laboratory data, and the development of computational models, environments, and assistants for designing. These provide promising areas for cross-disciplinary work, perhaps beginning with faculty seminars. Athena should foster them and monitor the full range of faculty initiatives in order to surface other such opportunities.

4. The faculty members who have chosen to devote significant parts of their lives to Athena-supported ventures need several things. They need technical support in order to develop software and they need the support of a well running and well maintained network. In addition, faculty who have become interested in educational issues raised by their projects need structure for reflecting on their experience. To provide the first, Athena should make a commitment to stabilize its technical system for a reasonable interval. One way to provide the second is to help developers of Athena projects conduct "self studies" of their educational initiatives. Two such studies, one completed, the other under way, may serve as prototypes. One is a study with Professor Jean Bamberger, conducted by visiting scholar Giovanni Lanzara, which documented the process of development of Music Logo and students' and faculty's responses to it. In the other study, Brenda Nielsen, a member of our research team, has joined with Professor Joseph Ferreira of the School of Architecture and Planning to explore student reactions to his LOTUS-based course on urban mapping and modeling.

5. The system of incentives in the departments discourages faculty from putting their "right hands" to Athena's work. But these incentive systems, focused on specialized disciplinary research and publication, are central to MIT. Changing them, even at the margin, is bound to be extremely difficult. But, if we seriously intend to pursue Athena's purposes, we must address this difficult problem. We will need to ask how to enable departments to evaluate the kinds of research that Athena's projects demand and to consider the design of career paths for those who turn their research to educational computing. In addition, we need to create, across departments, intellectual communities for young faculty who envisage careers in this emerging intellectual discipline.

6. Problems arise when educational computing blinds itself to questions of personal style. It leads to frustration, a sense of mismatch between students and the material, and to students having the feeling that they don't think "the right way." When educational computing ignores style, it sidesteps the need to develop systems that will facilitate the work of bricoleurs as well as planners, and to teach them in a way that encourages each individual to find his or her own path. There

are no simple solutions. But an appreciation of the importance of personal appropriation in how people use computers is a first step. It can insure that some groups of users don't get their wrists slapped if they approach programs in a different way than that anticipated by the software designers. Our case studies make it clear that it is naive to launch large scale experiments in educational computing that expect diversity in content but not in the form and feeling of computer use.

* * *

What follows are our four case studies, presented in the order in which the field work was carried out: Architecture and Planning, Civil Engineering, Chemistry, and Physics. Each case study is written as a self-contained document. We believe this adds to their usefulness at the cost of some slight redundancy. The cases touch on a set of common themes, although with different emphases, reflecting what was special in each of the settings, as well as evolution as we learned more about educational computing at MIT evolved.

The field work—observations and interviews—for the case studies of Architecture, Chemistry, and Physics was done by Brenda Nielsen and M. Stella Orsini; that in Civil Engineering by Wim Overmeer.

The authorship of the final chapters as they appear here is divided among members of our research group. The Overview was written by Sherry Turkle and Donald Schön; Architecture and Planning by Sherry Turkle, Brenda Nielsen, and M. Stella Orsini; Civil Engineering by Wim Overmeer and Donald Schön; Chemistry by Sherry Turkle and Brenda Nielsen[5]; and Physics by Sherry Turkle.

The case study work and final reports would not have been possible without a grant from the MIT Provost's Office to the Project Athena Study Group and without the support of the Program in Science, Technology, and Society which freed a substantial amount of Professor Turkle's time to work on the Athena materials.

The authors of this report would also like to thank Margaret MacVicar for her guidance though many stages in the development of this Project; Jean de Monchaux and the Project Athena Study Group for many helpful comments that got this study off to the best possible start; Gerald Wilson and the Project Athena Executive Committee for constructive debate on important issues; and

[5] The authors gratefully acknowledge the contribution of Lisa L'Hote.

Steven Lerman, Director of Project Athena, who from the very beginning of our work was generous with his time and expertise and who helped us through his thoughtful and challenging criticism.

Project Athena at MIT

by

Professor Sherry Turkle
Professor Donald Schön
Brenda Nielsen
M. Stella Orsini
Wim Overmeer

-gave me permission to use report

May 1988

Please do not quote or reproduce without permission.

Project Athena at MIT

Table of Contents

Project Athena at MIT: Computing in the School of Architecture and Planning

by Sherry Turkle, Brenda Nielsen, and M. Stella Orsini

May 1988

Introduction

Overview

This study of computer use in the School of Architecture and Planning underscores two kinds of impact. We see effects of the "instrumental computer"—the computer as it does things for designers and planners. Beyond this, we see effects of the "subjective computer" which follow not from what the computer does *for* designers and planners but from what the computer does *to* them as educators and practitioners. As a tool that serves as an extension of mind, the computer provokes new reflections on thinking. As a tool that impinges on professional life, it provokes a new look at the knowledge behind practice.

In this study we see how the computer affects people's way of thinking about themselves and their work. Its introduction presents a range of challenges to traditional models of the design professions, revealing how delicate is their balance between intuition and technique, between art and science. What remains when the designer no longer needs to draw? What remains when the architect no longer has to imagine a building but can see it projected in three dimensions? What is the essence of the designer's gift if it is not drawing and imagination in any simple sense?

In design, the computer is not only an "evocative object," compelling a new look at knowledge and practice but is also a projective medium of considerable power. This report highlights the fact that watching designers and planners use computers provides a window onto different styles of creativity in these fields. The study of computers in the education of design professionals is more than the story of a new pedagogical aide: one of our central findings is that the use of computers in the School of Architecture and Planning has had the effect of making larger questions about professional practice, creative style, and design philosophy in these disciplines more "on the surface"—more salient and discussed.

A first section describes the Athena project in the School. How have the machines been used? What were the intentions of the faculty in undertaking their educational experiments? How do they compare to students' account of what has happened? These considerations lead to a discussion of the differences between students' and faculty's view of "computer literacy."

A second section is more interpretive as it looks at the question of effects, in particular, the impact of computers on the learning process. It examines students' and faculty's opinions of what has been gained and lost with computer use, including how the use of these machines affects the way they feel about their work.

A third section explores an effect of the Athena experiment that goes far beyond the technical: the creation of a computer culture within the School. Computers became an everyday artifact and have catalyzed a new round of heated discussion about the changing nature of the design professions. Different people approach these changes in very different ways.

A fourth section goes further in exploring diversity of response to the computer. Athena's strategy was to let a "thousand flowers" bloom, by encouraging a diverse group of faculty-initiated experiments. These experiments were expected to differ because faculty would develop different software and teaching methods. Our study of Architecture and Planning underscores another way in which you get many flowers within an educational computer culture. We found differences not only in what different faculty offered to students, but in what students made of it. Students developed highly individual styles of using the computer. This personal appropriation of the technology seemed a significant element in successful learning. Students make the computer their own in their own way, a fact with important implications for educational policy.

This report ends with a "Coda on Technological Determinism." As an educational experiment Athena began with a gift of hardware. The "Athena culture" as it has developed within MIT departments compromises educational goals because, perhaps unintentionally, it perpetuates a value system which continues to put technology first.

Methodology

The goal of the larger study of which the School of Architecture and Planning is one case was to understand the educational impact of Project Athena, including social and psychological aspects of introducing computers into learning environments. Thus, our interviews with faculty and students in the School cast a broad net, including discussion of personal experience with computers as compared with other tools, concepts of computer literacy, perceived advantages and disadvantages of using computing, and the implications of the technology for design education and the design professions. Our interviews, using guideline questions and an open-ended format, allowed us to collect comparable data from each

respondent while permitting flexible discussion of topics of particular concern to specific individuals.

We interviewed eleven faculty members in the School of Architecture and Planning, all involved with computers in their teaching and research, who represent the mainstream of the School's Athena-funded computer work. For student interviews, we randomly selected fourteen students, seven men and seven women, from those enrolled in the major computer-related courses. They included graduates and undergraduates from both the Architecture and Planning programs.[1]

For the purposes of this study we treated "Architecture and Planning" as a single unit. But it is helpful to make a distinction that essentially sets designers apart. For architects and those planners involved in design, the introduction of computers takes place in a highly charged atmosphere. Computers provoke fundamental questions about the nature of design talent and the relationship of the designer to his or her work. For those planners who use computers for modeling and spreadsheet simulations, the issues seem less emotional. Planners report that the computer changes the nature of their work experience (for example, by offering a window onto the process of model building), but it does not seem to challenge their sense of professional identity as it does for architects.

The Model of Evaluation

A simple model of educational evaluation assumes that an innovation has an impact and that the members of the effected community have come to a position about it, i.e., that it is more or less good or bad. Then, the job of evaluation is to ascertain and sum these positions. This case study challenges any such model. The individuals most closely involved with Athena are ambivalent. To illustrate, take the example of their mixed feelings about using computers for drawing. They report that they use computers to compensate for lack of drawing skills (this means that a wider range of students can participate), but the results have a lower artistic value. They can try out a larger number of solutions with computerized drawing, but it is easy for students to get lost in the multitude of options. When they use the computer they can change their design more easily,

[1] Most faculty interviews were conducted in one session, varying in length from one to three hours. Student interviews were of roughly the same length, but were usually broken into two parts: the first dealing with general issues and the second with specific questions suggested by the initial discussion. In addition to personal interviews, the computer-related courses most referred to by faculty and students were observed, facilitating informal interviewing of additional faculty and students.

but they feel a loss of 'attachment' and commitment to their product. The computer speeds the revision of drawings, but their product lacks the personality of hand-drawn work.

The conflict of feelings about drawing are typical of how designers talk about the computer presence and thus the Athena presence. Here we try to portray their ambivalence. It illuminates the issues that must be faced by introducing computers into education and practice in these fields.

Part I. Learning in the CRL

The Background

Pressure to introduce computers in the School of Architecture and Planning came from forces external to MIT. In the course of the 1980s, it became clear that graduates needed to be able to use computers to successfully compete for jobs. Before then, computer use in the School was extremely limited. There was timesharing access to MIT's mainframe computers, but this system was of limited use to architects and planner. The cost of setting up a facility tailored to their needs seemed prohibitive. The existence of Project Athena was decisive. Its resources made it possible to create a computer laboratory.

The first laboratory consisted of a few Osborne computers in a spare room in Building 7, but it grew quickly with the addition of nine IBM XTs from Project Athena in the spring of 1984. In the fall of the same year, an Autocad system and peripheral equipment were added and the laboratory moved to the Emerson room in Building 7.

The bulk of equipment from Athena's general allotment arrived in the winter of 1984-85; eighteen IBM ATs were delivered. This core of machines plus four additional ATs acquired in the summer of 1985 was further supplemented by the School's purchase of numerous digitizers and printers. The lab itself, known as the Computer Resource Lab or CRL, is now housed on the fifth floor of Building 9.

The sign on its door reads "THE GARDEN Computer Resource Lab." This name celebrates and is celebrated by the ceiling height trees in the center of the room and reflects its atmosphere—informal and playful as well as reflective. The

Garden is open twenty-four hours a day and the vast majority of Architecture and Planning students know the combination for the lock to enter its main room.[2]

From the first, the CRL has been an experimental environment to explore computer applications in design, planning, and architecture. It is the laboratory setting for the Athena-related courses the School offers (approximately 10 per year). Of these courses, some integrate computers into existing curricula. The majority, however, are new courses, focused on computer applications. Two of the courses teach programming; the "technical" content of the others consists of learning to use off-the-shelf software packages such as spreadsheets and computer-aided design packages.

The Garden is one of the most heavily used computer facilities on campus. Architecture and Planning students use its equipment for personal use and research projects as well as for work in both Athena and non-Athena courses. The computers in the Garden are used most often for word processing, followed by spreadsheets, computer-aided design and graphics applications.

The Garden is not quiet, but animated. Conversations about technical and personal matters—and expressions of frustration at machines—are carried on aloud. Students appreciate the lively social life of the Garden, but there is demand for more private space. For example, some students mentioned that in this noisy environment they have some trouble concentrating, particularly on writing.

In the Garden has grown up a style of mutual support. As one student puts it,

> I think it's part of the learning process, to meet people, exchange information and see what the other person is doing. Otherwise, one would have to take a course for every program application. Most people learn by looking at other people in the cluster. The information you get in class is minimal.

Help is casually and often spontaneously given to obviously frustrated users; students known to be experts on some aspect of computer work are actively sought out by those who need help. Patience in giving help is a characteristic of the Garden culture as is a norm of politeness in the transfer of a place at a computer to the next scheduled user.

Plans for the future of the CRL include the development of additional networking and multiple-task capabilities as well as the introduction of higher resolution displays and increased possibilities for combining graphics and text. The fac-

[2] The Garden has a second room where the computers with CAD systems used in courses are housed.

ulty who run the laboratory hope that the expertise acquired on the systems now in operation will make it easier to move towards larger machines and increasingly complex networks. They see the CRL as the first step toward a more visually (because computationally) rich design environment in the future.

The Challenge of the Computer: Faculty and Student Perspectives

In the Department of Urban Planning the faculty members most directly concerned with Project Athena are department head Professor Gary Hack, Associate Professor Joseph Ferreira, Professor Karen Polenske, and Steven Ervin, teaching assistant for Professor Hack.

In the Department of Architecture, the most involved faculty are acting department head Professor William Porter, Associate Professor Patrick Purcell, Assistant Professor Frank Miller, and Peter Jorgensen, a graduate student who teaches computer-related subjects.

Professor Joseph Ferreira, the director of the Computer Resource Laboratory, teaches courses in statistical analysis and mathematical modeling. He believes that model building teaches general problem solving skills. In his course, Ferreira stresses experimentation with a variety of models and kinds of computer environments. He sees the computer as a way to do numerical calculations with speed and precision, but more important, as a means of "unlocking the black box of modeling." The computer can change the way students think about planning because it allows them to do more than perform mathematical functions. They can have a better sense of what the functions mean.

Professor Ferreira thinks that the CRL has two related goals: to provide students with state-of-the-art equipment for time-saving operations and to provide an environment for a more in-depth understanding of computing technology and its potential. He believes that the computer facilitates a new kind of "learning by doing." Like other faculty, his focus is on general fluency

> not so much on the mechanics of spreadsheets or in running a particular model of a spreadsheet, but in having to construct, change, interpret, or understand the way in which spreadsheet models work, and then to use that to understand more about modeling and ... about computing environments that are flexible and powerful.

Professor Frank Miller, in Architecture, is also concerned with the nature of insight in design. He is exploring the potential of computer graphics systems like Autocad to help students develop the intuitions about design that he feels are

central to creativity. In Planning, Professor Gary Hack's major interest is site planning. He teaches it through projects where students have to use what they learn in concrete situations. Like Professor Miller, he feels that students need intuition or "designerly insight" along with the more specifiable "rules of thumb." And like Miller, he believes the computer may help. Steven Ervin, teaching assistant in Professor Hack's site planning course, has made the computer his main professional interest. He, too, believes in the technology as a way to develop intuition as well as skills.

All of these faculty see the computer as a means of enhancing the experience of design and planning. They expect it to do more than perform an instrumental role—they expect it to change the way their students think. In their vision of a visually and conceptually richer design environment, students need a broad fluency with computers. This faculty has little interest in the computer as a purely technical object, and this translates into a reticence to use class time to teach computer basics. Professor Porter admits that he depends on students learning about computers by "picking it up on the outside." Professor Purcell allows that programming is required in his course, but it is not his focus. Rather, he tries to give students a "feeling for what programming is." Professor Miller expresses a similar desire to have students move quickly beyond the technical to get a "feeling" for use. His approach is typical:

> I wish I didn't have to spend any time talking about how to use the machine. I have to because no one knows how to use it .... I give them a very practical training on the Autocad system. they use only a third to a half of the commands. They don't need any more for what I am trying to do.

Faculty teach the minimum of practical skills and this is where a tension between them and their students begins. Faculty are interested in having students develop a broad picture of the computer's role in architecture and design. Professor Purcell puts it well. He speaks of a "critical position" or "critical stance" toward the use of computers in the field as more important than being able to use any particular computer facility or application. Faculty are concerned with the computer's relationship to particular design issues and don't want the computer *per se* to compete for students' attention. In general, faculty feel that the less conspicuous the computer, the more directly they can focus on the "real" subject.

Students are in a very different position. They must work under a constraining pressure: in order to pass their courses, they must master the instrumental use of the machine. And they must do so in an environment that leaves them to do it by themselves. Students lack the faculty's freedom to philosophize. They

do not have the intellectual leisure. Indeed, they report that they have to spend much more time than they expected (or that the faculty seems to expect) learning to use the computer.[3] Even mastering the basics demands a heavy investment of time and energy. Faculty and students sometimes give the impression of talking past each other; students are hungry for a kind of information of which faculty tend to be dismissive.

Students need computer skills to get through their courses. They have good evidence that they will need them to get good jobs. But many feel that the amount of time they spend developing them detracts from their professional studies. Most wish that faculty would make it easier for them to learn the basics so that they could move more quickly to applications. Students feel under pressure; it comes out in their concern with budgeting their time and in their way of talking about computers that stresses "keeping them at a distance."

Students emphasize the need to carefully gauge the amount of time spent on computers in order to keep them in their place. At all levels of expertise, they are wary of becoming too involved. They are searching for perspective, picking up on the faculty's interest in the "big picture." Students believe that faculty may have the perspective they want, what Purcell called a "critical stance" toward the machine. They think, somewhat naively, that this perspective is something in which they can be instructed. They describe it as "having the right answers" about the computer. And they often feel that faculty are unwilling to share these answers with them. Students, therefore, are doubly frustrated. First, they feel alone in their efforts to learn the technical machine; they are frustrated about skills. Second, they feel at a loss about how to acquire a sense of perspective about how computers fit into the larger professional picture. In the end, students pick up the technical details of computing on their own. And their discussion of where computers fit into the "big picture" largely takes place outside of the classroom.

Faculty understand student frustrations. But, they fear the cost of meeting student demand for what they see as "cookbook instruction." Professor Ferreira describes explicit educational goals in designing the CRL: first, dynamic rather than rote learning, and second, a support system for learning outside the classroom. He, like other faculty, remains committed to these goals despite faculty

[3] MIT tradition favors teaching programming as an aside to courses. Even the main programming course, 6.001, teaches the LISP language "indirectly," and focuses on larger issues in computational structure. This strategy of expecting students to indirectly acquire computer skills may work within the Electrical Engineering Department, but the verdict is less clear in departments such as Architecture and Planning.

sensitivity to student frustration.

Faculty are in general aware of students' problems but they embarked on this Athena experiment with no plan for how to be more systematically in touch with student experience. So, for example, Professor Purcell feels that evaluation is not his area of expertise—"my job is to design and implement the best possible computer project I can." His expectation was that "someone would come in" and evaluate the project at the appropriate time. His perspective is typical of his colleagues. But the fact that students and faculty are having such a different experience of the computer suggests that this widely shared attitude about evaluation may be a problem. Continued self-study rather than the punctuation of a one-time external evaluation seems more appropriate to the needs of the group.

Meeting the Challenge

Most Architecture and Planning faculty share an ideal model for teaching with computers. It has three phases: first, students are presented with problems in architecture or planning that call for using the technology. Second, as they work, students are expected to analyze design and planning issues as well as the computer's utility and limitations in the particular application. Finally, there is a phase of generalization in which students are expected to come to a "critical stance" about the role of the computer in Architecture and Planning.

The faculty describe many constraints that make this ideal model hard to follow. There is little time for dealing with technical issues and there is concern that students will be intimidated by "too much philosophy."

> We haven't dealt with critical issues in class. If people get overly self-conscious about their process, they can't really be involved. They can't be doubting and doing at the same time.

So, in practice, formal instruction does not center on either technical skills or broader issues. If the ideal model has three phases, phase two and three are given short shrift. What ends up being emphasized in class are the substantive aspects of the exercise phase. The responsibility for skills and for generalization falls to students. But since there is such a heavy premium on the technical knowledge—students need it to do the exercise—it really is what gets students' time and attention. While for faculty, computer literacy means a "critical stance" towards computers *in general*, for students computer literacy has become synonymous with technical master of the Athena system *in particular*.

For students, computer literacy has also become synonymous with feeling "in control." "If I felt more confident with the program and felt more in control of it rather than being at its mercy, so to speak, I would feel that I could use it more ... and decide when not to use it." Given that students are expected to acquire technical mastery on the side, their options are limited. It is not possible for most students to achieve a feeling of control through deep understanding. So, they compensate by developing a particular style of using the computer. It gives the feeling of having technical mastery and control because it feels personal and direct. We call it "customizing."

In the computer world, the word "customizing" refers to modifying the functional characteristics of a piece of software. But when these students talk about customizing, they usually do not mean that they are changing the software, but rather that they are making it do something that is both personally desired and that goes somewhat "against its grain." For example, a student will take a computer graphics system that was designed to make very precise drawings and try to make it do rough preliminary sketches. The feeling of personal control is achieved not by changing the "innards" of the software, but by using it in an unorthodox way.

In many ways the impulse to customize is familiar to designers. Professor Hack describes the designer's tendency to create new forms by taking traditional materials and using them in idiosyncratic ways.

> It's like giving an artist a precise tool like a pencil to sketch with. They'll use it sideways, rub it along the paper—and that's what artists do and what creative discovery is all about.

But the customizing we see in the current Athena environments has another and less positive side. When the computer is used in the designing process, it is an instrument of individual expression, an "intimate machine." Its users want to personalize it in order to achieve what several refer to as a "sense of oneness" with it. They want to feel that they "own" it. But if you do not have the technical knowledge to actually tailor it to your ends, you look for idiosyncratic uses to give you that feeling. In this sense, the current brand of customizing is a response to frustration. It gives an experience of the computer as "yours," as though you have your own way with it.

Customizing as a way of feeling in control with limited technical ability has a cost: it puts the technology and playing with its limitations at the center of students' concern. It puts a premium on a style of "hacking" that is not tied to any articulated educational goals. Customizing is a way students use to make comput-

ers more comfortable. It is expressive behavior and it is important for faculty to correctly read what is being expressed—a desire for control, a desire for personal appropriation. It points to the students' struggle to make the computer their own despite limited technical resources. Greater comfort and confidence with the machine could allow for more productive forms of personalization. While the impulse to customize is natural and desirable, more technical knowledge would allow it to be realized in a richer because more real way.

Part II. The Computer As Change Agent In Learning

With different degrees of sophistication, students in Architecture and Planning are becoming competent computer users. They are taking the courses, doing the assignments, completing the projects. But evidence that there is a growing core of computer-sophisticated students does not address a larger goal of the CRL: the integration of computers into the learning process. Has the computer served as a change agent for learning about design and planning? Has experience with the computer affected students' disposition to use or cultivate their capacities? Has it affected their sense of ownership of the artifacts they produce? Has it changed their feelings about their practice and their future professions? Does the computer make some things easier? More difficult? Or impossible? Does it encourage a focus on certain issues and a selective inattention to others?

Here we describe the opinion of faculty and students about such effects through their report of experiences with and without the computer. Although opinions cannot substitute for direct observations, the opinions reported here seem to us to have particular credibility because they are not "partisan." For example, people with strong pro-computer beliefs also articulate the "down side" of computer use in its current embodiment in the Athena context.

We begin with a report on the perceived gains of using computers in design and planning, and then report the perceived losses.

What Is Gained

* Social Interaction

In general, faculty think of computers in education as having the potential to change the way people learn.[4] They see computers as a medium that has brought people together. Faculty speak of students "knowing each other's projects" in a more natural way and about "knowing not only what other students do but how they do it." This high degree of student-student interaction was part of the explicit design of the CRL. Involved faculty vetoed an original Athena plan to construct partitions in the lab and the CRL has a policy of hiring relatively inexperienced students to work in the lab as tutors. Professor Ferreira reflects that although "this is not the most efficient system," it increases the number of students who are socialized into the CRL culture. Faculty stress their success here: the CRL culture is an active, social, and collaborative learning environment. But the changes in learning that faculty attribute to the computer presence go beyond these social affects.

* Challenge to Traditional Practice

The computer becomes an agent for change by provoking questions about current practice and traditional techniques. Professor Hack talks about how computers call into question the use of contour lines to represent changes in elevation on site plans. "Perhaps someday," he reflects, "we will find contour lines a very primitive way to represent terrain. We are used to them because we have 'always' used them, and because if all you have to work with is a pencil, making lines is a perfectly natural convention." But computers liberate the visual imagination: "instead of the aerial view we'll possibly shift to a worm's eye view." The computer provokes a re-evaluation of current methods and points to new directions. And Professor Purcell believes that students who work in electronic environments "where information is stored in many forms and in many locations" develop rich cognitive models of the computational environment which enhance their general learning ability. Professor Ferreira has worked on the use of Lotus spreadsheets to teach modeling. For him, computers represent a new vista, making it possible to "open up the black box" of a model and develop a deeper understanding

[4] Professors Purcell, Ferreira, and Miller express strong convictions about the computer's potential as an agent of change in education. Professors Hack and Porter are more reserved. Although not yet convinced, they too are interested in the computer as more than a way to do current jobs more efficiently. They too are interested in its potential for allowing new kinds of learning.

of its internal dynamics, leading to a more profound understanding of theory. In general, the computer presence is given credit for encouraging creativity and providing deeper insight into process.

* Creativity

Most faculty members agree that the computer provides students with new material for innovative solutions to design and planning problems. They believe that the computer enhances creativity in several ways. First, with the computer one actually 'does' design differently. For example, drawings can be built by aggregation—the grouping and repetition of elements rather than line by line. Professor Hack cited one student's experience as exemplary of the novel outcomes he has come to expect. Having discovered in a magazine a picture of a residential cluster of houses that appealed to him, this student used the computer to proliferate the units and combine them in larger modules, a process he likened to moving from leaf to branch to trunk on a tree. The resulting design was highly original. In Professor Hack's words: "There aren't many street layout patterns that I haven't see before. There aren't many variations that will surprise me. But this one truly did."

Second, the computer provides a new perspective on your work. Professor Miller believes that computers help students get unstuck when they reach an impasse because it allows quick changes "on the order of magnitude of turning your paper upside down." Miller feels that the possibility of quick and low-cost revision liberates his students to take advantage of their mistakes because now they generate less anxiety. In addition, reflection on computer-induced mistakes can lead to seeing or doing things in new ways.

As students, working with computers, learn to take advantage of trying alternatives and reworking mistakes, the less attached they are to particular choices. Miller explains, "You get used to shoving around large amounts of stuff and being willing to risk the design progress you've made in order to get further." Professor Hack also observes that computers make students more willing to experiment and more willing to give up on ideas that don't pan out. His students come to a seeming solution and say that it is not necessarily the best or only one: "It makes it easier to move on to other alternatives." As a result, says Hack, computers make it easier to criticize students' work and ask them to try other possibilities.

* Insight Into Process

Faculty credit computer materials with providing enhanced understanding of design process. A computer program requires its user to be explicit about his

or her decisions. Professor Porter describes the positive effect of this heightened awareness. In planning for a shopping center,

> You can give it a range of locations and it might choose different places for you, depending on how you wrote the program. The student must ask: "What is the empirical meaning of step one?" It forces students to think about that as a question. They start thinking very clearly about modes of transport, and about relationships and priorities of locations. They begin to think in a dynamic way about how urban systems work. It gives them a chance, as designers, to think of ways to conceptualize a city that can be embedded into a computer system.

For Professor Porter, the ability to "relate decisions to empirical reality, to see their correspondence to real processes" is of paramount importance because "the more you recognize the fitting or not fitting of your decisions to the real process out there—and can apply that knowledge consistently—the more likely you are to be successful."

Other faculty claim that their students are both more analytical and more intuitive when they use the computer as a modeling tool.[5] Professor Ferreira believes that the computer allows his students to move beyond the rigid application of models toward a more flexible approach to generating a range of appropriate solutions. For him, spreadsheets are "useful metaphors for teaching modeling concepts and forcing students to think about how they'd like the computer set up for specific purposes." Ferreira believes his students have moved from these specific understandings to broader insights. "This is an important skill—planners tend to be involved in problems for which there is much less theory about what's right and much more need for invention."

What Is Lost

* Attachment to the Product

Students agree with faculty that the computer's capacity to make multiple iterations so easy has important advantages: one moves rapidly through a series of alternatives without becoming overly attached to an individual solution. The user is freed up to be more creative. But faculty think that there is such a thing as *too much* detachment. They see evidence of this in students' work and speculate

[5] Indeed, computer use softens the perceived boundary between what is analytic and what is intuitive.

that the computer contributes to it by moving design from paper to screen which implies a change in the emotional relationship to the material, a loss of personal symbology. Professor Hack elaborates:

> You love things that are your own marks. In some primitive way, marks are marks .... I can lose this piece of paper in the street and if [a day later] I walk on the street and see it, I'll know that I drew it. With a drawing that I do on the computer, a digitized map, a square or a circle on the screen, I might not even know it's mine.

Professor Hack believes that the relationship between design practitioners and their products, a relationship of great intensity, is now compromised by the computer. The computer gives a knowledge of one's product that does not inspire "love." "People do analyses of their plan [on the computer] but they only fall in love with the marks they make themselves." Students, too, comment on the loss of personal artistry in design.

> I work freehand a lot. I don't want the thrill of working freehand and seeing the product then translated onto a machine. Each person gives some kind of character to their work. Working on the computer may destroy that.

The computer allows people with poor or limited drawing ability to get involved in design. It does the drawing for them. But since "drawing is a way of knowing," people without graphic skills may never come to the intimate knowledge of their own work that results from drawing. A connection between the designer and the design is compromised. The situation is, at best, paradoxical.

* Attachment to the Real

A computer presents the user with a representation of the world on a screen. Professor Hack believes that the "screen as intermediary" causes the practitioner to be more detached from the real than when the world is represented in a drawing or model. Hack believes that this detachment is due to several factors: There is of course the fact that the screen is small, but beyond this we have particular habits of mind about "the way we think of the screen, probably related to our 'watching' television." We think of a screen as something we observe with a detached rather than participatory stance. We do not "own" its contents.

Deeper cause for detachment from the real lies in the process of computer-aided design. For example, when doing site planning, Hack explains that

> Students can look at the screen and work at it for a while without learning the topography of a site, without really getting it in their head as clearly as they would if they knew it in other ways, through traditional drawing for example ... When you draw a site—when you put in the contour lines and the trees—it becomes engrained in your mind. You come to know the site in a way that is not possible with the computer.
>
> It's terribly hard to manipulate images on the screen whereas if you have a plan of the site it is very easy to quickly lay over a piece of tracing paper and to quickly just take that thing and make some marks on the page. [Here, Professor Hack takes up a piece of paper and scribbles on it to illustrate his example.] That's taken me all of ten seconds to do; for me to do that on even the best of our programs at the moment is a five minute job. So by that time, I've had to very deliberately think of what I'm doing and convert it into a much more rational process whereas here all that I've done is said 'somehow when I see these slopes here ... they could connect that way.' In other words I can make an imprecise statement quickly on paper but not on a computer.

He goes on to elaborate his argument about detachment by telling the story of a student who was working on a site that Hack knew very well. The student drew a road on a slope that Hack knew was too steep to support without undertaking major land-moving and major expense. When Hack asked the student to explain his design, he replied, "Well, it's only one contour."

> It was only one contour, but that contour was twenty-five feet. The computer had led to a distortion of the site in the student's mind. He couldn't put more contour lines on the drawing, he said, because it was "too confusing." He said, "I couldn't work with that many contours ... I can't tell the lines apart anymore."

Given the scale of representation on the computer, if the distance between contour lines was assigned a value of less than twenty-five feet, there would be so many lines that they would not be distinguishable. So, in part, the problem lies with the limitations of current technology: the screen was too small and there wasn't room for more lines. It is possible that better technology will correct the tendency for this kind of problem. But perhaps the point of Hack's story transcends the particular case. Perhaps we shall always have a tendency to give "screen versions" a truth that they should not have.

* Aesthetics

There is evidence that in the Athena computer-intensive courses, the sense of accomplishment does not come so much from pride in the final product as from having gained insight into design process. In most studio courses students prepare very finished-looking drawings as part of the presentation of their projects. But in computer-based studios, such as Form Processing, the final products tend to be unattractive and often banal. Many students say they are disappointed in their end product, but make positive reference to insights, to what they have learned about their creative process.

Faculty members recognize that in Athena courses a new balance is emerging between process and product. In evaluating student work, they tend to stress process rather than the aesthetic quality of finished projects. There are times when some faculty find the aesthetics too demoralizing and encourage students in artisanal "compensations," for example, using pencils to enhance the appearance of computer projects. There is no simple answer to the question of whether in the end students feel more or less sense of accomplishment. They feel less pleased according to old criteria; but they seem to be developing new criteria for what accomplishment is about. There is, too, a new ambivalence, not just about the less-than-beautiful object, but about the emerging shift in criteria for excellence.

Architecture students try to compensate for a sense of "design loss" by allocating their time in a way that makes it clear that their first allegiance is still to art. They are concerned about whether they are abdicating too much control over the process of design to the computer. "It is easy to let the computer model the way you perceive things," says one. "You can let it manipulate you instead of the other way around. And then you've lost the game." Traditionally, mastery of all aspects of design process (including, of course, design technology) was a hallmark of professional competence. For many students, the computer calls this sense of mastery into question.

Part of the reason that software for computer-aided design seems manipulative is that it demands that problems be solved in relatively rigid sequences. It is possible that technical progress will make the designer feel more autonomous when he or she is working with a computer. But feelings about mastery may be challenged by the computer *per se*, not just the currently available, primitive form of what Athena now offers. Professor Miller reflects that "each CAD system has its own epistemology, its own way of structuring the world. It is a much more deliberate and complete epistemology than pencils and tracing paper because it was in fact structured on a certain point of view." And he explains that "the software

designer's point of view is never my point of view and my point of view is not the next guy's." We saw how students and faculty attempt to soften this loss of control through customization. Another strategy for feeling in control is to use optimism about future generations of computer-aided design that will allow the personalization they desire.

* Transparency

Some planning students, the group of non-design oriented planners, are less concerned about these questions of personal style and epistemology. Planners use software packages for word processing and statistics—applications where there is less danger of getting lost in "the epistemology of the machine." Planning, at least as these students think about it, isn't the kind of enterprise that will be fundamentally changed by computers. Planners are therefore less anxious than architects about adopting the new tools. If the computer makes them more efficient it is not controversial. This is not to say that planners are without some ambivalence about computer use.[6]

For example, Professor Polenske enjoys using the computer in her economics courses because "more classroom time can be spent thinking about what the numbers mean." But she also sees a negative side: there is a loss of transparent understanding.

> When I learned statistics we had to do regressions by hand. You knew exactly what was happening. When I inverted my first matrix it was done by hand. Two of us sat down with a calculator. It took us an hour to invert the 3×3 and to check it, but I knew then how a matrix was inverted.
>
> Today, many people do a calculation without really knowing what's happened. I see that in my classes—in the interpretation people give to an impact. If they'd worked through it, seeing each of its iterative stages, they would have had a much better understanding of what direct and indirect impact really meant. When you question them as to what the terms mean, after having used the computer, they don't know.
>
> The summer before we used the computer for my course for the first time, I had a student working on a shift-share analysis. He was delighted after working for a couple of days. He said,

[6] The distinction between Architecture and Planning suggests a typology of disciplines; some fields—attached to particular ideas about process—have more reason to feel that the computer will challenge something essential in their nature. The distinction between Architecture and Planning also suggests a close coupling between a discipline's attachment to its material culture and the level of computer anxiety within it.

> "Gwen! Gwen! Come look at this." He pushed the button five times and out came the correct answer. I said, "Bob, what in the world will those students learn through this?" He was so delighted by the degree to which it could be automated! But what would students learn? It has lost the end purpose of what the exercise was supposed to be. He adapted it quickly to something where the student had to think through some steps and push buttons only on mundane parts.

Part III. Computers and Changing Images of the Professions

Design and Planning as Computer Cultures

The creation of the CRL has changed more than the content of courses. It has encouraged the creation of a computer culture within the School. Becoming a member of this culture has become part of students' professional socialization.

Specifically, the presence of the CRL has contributed to students feeling that "computers are everywhere." They feel left out if they don't get involved with computers. Mark, an undergraduate in Architecture, complains about the pressure to use computers "even if only to write your resume. If you have access to a word processing system and a laser printer, you feel stupid if you get it typed. You feel dumb, behind."

The visibility and prestige of the high-technology group within the School communicates a message that "computers are the cutting edge." Monica, a graduate student in Planning, had a typical reaction: "In the Planning program we take computers for granted .... Students must become familiar with computers; they are part of our society, soon they'll be like phones." Roger, a graduate student in Architecture, expresses a similar thought but with somewhat greater anxiety: "If architects aren't able to bring computers into the scenario of the design process, they will become draftsmen. Maybe that's too rough, but that's the idea I have." He interprets past crises in architecture as due to architects not getting involved with "technical stuff." "There are so many components to modern structures," he continues, "that you have to use a computer to design them. If architects don't do it, they will once again be overcome by civil engineers."

In sum, students feel that computer competence is part of the required package for professional practice. It is required to "get a job," to "talk to people in

the profession," and further down the road, as one confident student explained, to "talk to my employees, to ask the right questions." Students feel that not only do they need to be able to use the computer but in addition, they need to participate in the debate about the future role of computers in their fields. Apart from *using* the computer, students expect to be taught how to *talk* about them.

Coping With the Computer: "Just A Tool"

Faculty and students believe that computers have the potential to revolutionize design and planning. But the computers they currently have to work with fall far short of such big things. They are relatively clumsy instruments, ill-adapted to design. Faculty and students reconcile this tension by adopting two parallel ideologies about computers in their profession.

A first defines the computer as "just a tool." A second projects a future for computing in design and planning where the computer, far from being "just" anything, becomes a central actor in the creation of new epistemologies. People hold both of these views at the same time. Indeed, holding both together serves several useful functions. The minimalist position allows people to accept limited mastery over a limited tool in the present while the maximalist position leaves the path open for a more significant relationship with computers in the future. The two ideas complement each other; together they provide a reassuring framework. The disappointments of the present are rationalized; the future is idealized.

Holding both the minimalist and maximalist views serves another function as well. People recognize the potency of this technology. Already, computers are changing design process. It is exciting, but also disconcerting. Architects in particular fear what the computer may ultimately do to their profession. So there is pressure to keep the computer in its place until the consequences of such changes become clear. Dismissing the computer as "just a tool" is a way to do this: everyone can agree that computers are helpful for calculation, working drawings, and mechanical or repetitive tasks. The more an individual suspects that the computer may ultimately have a revolutionary and personal impact, the more the "just a tool" idea is evoked as an attempt to deal with the volatile nature of the technology. Anxieties about the long-term effects of the computer's creative capacities are dealt with by projecting them into the future.

Changing Design Epistemology

In that future, the machine will change the way that they think about fundamental issues in design and planning. Professor Hack summed this up: "Tools not only change the way you know things, but also create the potential for doing new things that you never did in the past." Peter Jorgensen, a doctoral student, captures this idea with an analogy: "Using the computer is not like making an automobile where they had to automate a horse and buggy. It's more like making an airplane where the laws and rules are different." Professor Hack is frustrated by current conservatism:

> We are trying to continue a paper and pencil tradition with a tool that doesn't lend itself to that. We need to find a new method ..., one that is powerful and engaging, more compatible with the machines' potential.

Faculty talk about how designers will no longer need cardboard models to study volume and massing, how advances in networking will extend the possibilities for information sharing, and how the ability to quickly transform design will extend the imagination. Students share these visions. They make matter-of-fact statements about the future of computing such as, "Computers will be an extension of the mind and influence how people think and perceive," and even, "Computers are superhuman; in the multi-tasking capabilities, they will go beyond what any one person can do." But new ideas are anxiety provoking as well as exciting. As we saw, the "just a tool" position is a way to deal with it.

Defining the computer as "just a tool" makes explicit the boundaries for the relationship with the machine. Seen as "just a tool," the computer remains under the complete control of the designer, subject to his or her command. Most importantly, when the computer is "just a tool" it is divested of any creative power of its own. Faculty and students don't want the machine too close to the core of design, seen as a creative act in the mind of the designer. To make it more tool-like, some suggest making it more opaque, so that it does not "require the user to know how to program or learn skills of operation." The less they must focus on the machine, the freer they feel they will be to concentrate their thoughts and efforts on creative problem solving.

Students peppered their expressions of reticence about the computers with phrases such as, "I will never be a computer hacker" and "I don't consider myself a hacker." Defining the computer "as a tool" is a way to reassure themselves that they are not even coming close.

Hacking and Anti-Hacking

Hackers have a strong image at MIT. Here we define the hacker as someone who loves the computer not just for what it can do, but for the machine itself. The cost of this passion: it is easy to become lost in the technical. MIT students see hackers all around them (including on the Athena staff) and many define themselves in opposition. Within the "artistic" disciplines of Architecture and Planning, the need to differentiate oneself as a "not-hacker" is particularly strong. One graduate student in Architecture explained how it is not possible to be both a "computer person" and a good designer at the same time.

> You don't have to be a very good programmer. No. Then you wouldn't be an architect .... Can you be a surgeon and a psychiatrist? I don't think you can. I think you have to make a choice.

Architecture is a combination of the aesthetic and the technical. For each practitioner, this combination is volatile. The computer has become a new element in the delicate balance of architectural art and architectural science.

Students feel pressure to be involved with computers but fear that too much involvement threatens to upset this balance. They see the computer as part of the science rather than the art of the field. Students, like faculty, are not sure at what point using the computer compromises art, but the anxiety is there along with a fear that the computer is too seductive. Even students who have become heavily involved with the CRL express this anxiety: the computer could compromise their professional identity.

Some students describe the tension between "being a good programmer and being a good architect" as though the technicity of the one impinges on the artistry of the other. Other students fear that if they become too competent, the balance of their practice will shift to programming: "I don't want to become so good at it that I'm stuck in front of a computer 40 hours a week. It's a matter of selective ignorance."

To think of the "computer as tool" idea as resistance to computers is too simple a model. Rather than hindering the adoption of computers, the "just a tool" rationale facilitates it. It quells current fears by circumscribing the computer's domain of influence. We found that talking about the computer as a tool allows architects and planners to experiment with computers today in spite of their reservations about where it will lead them tomorrow.

Part IV. "A Thousand Flowers": Styles of Appropriation

The starting philosophy of the Athena program can be captured in the phrase "A Thousand Flowers." Athena's strategy for educational innovation: let a multitude of faculty-initiated experiments blossom. And let the sturdiest of them survive. Athena expected new teaching methods and new pieces of software. But the history of educational computing in the School of Architecture and Planning suggests that another kind of diversity is emerging as well. Different people have very different styles of appropriating the computer. This has important educational implications.

There is a widespread notion (even among educated and technically sophisticated people) that the computer can only be effectively used in one way. Even though there is much talk about the versatility and flexibility of computers, this rhetoric usually coexists with a conflicting assumption that by its nature, the computer imposes a certain way of working. This is a planner's way of working (not to be confused with "urban planning"): rule-driven, formal, and proceeding from global to local.

This image of what the computer imposes has several corollaries: first, the assumption that computers are efficient tools for users who have natural tendencies towards the planner's style; second, the assumption that people who have a different way of thinking will either adapt to the machine's style or be frustrated in their computer use. The corollaries show up when people describe themselves and others as "computer types" or as "the kind of person who would not be good at computers."

The case of Architecture and Planning challenges these assumptions. There, we see a range of styles in students' ways of approaching and using computers. Diversity rather than homogeneity characterizes their appropriation of the machine. These styles are evident not only in what students do, their actual methods of work, but in their feelings about what they do. They show up in students' differing computational aesthetics: what they find fascinating, valuable, exciting, frustrating, and useless about the computer.

People make the computer their own in their own way—but only if they are allowed to. The School of Architecture and Planning is a rich environment to study styles of mastery because the faculty has encouraged personalized uses of the technology. They speak eloquently about the necessity for students to take an individual approach and arrive at personal understandings. They differ in the kind of knowledge they feel students need in order to develop this

understanding—their range of views reflects the fact that they, too, have different styles.

Professor Ferreira teaches the details of the operation of the machine because he assumes that students need to achieve a personal style of work and can only do this with a "transparent" understanding of the system. Professor Miller does not stress transparent understanding. On the contrary, he believes that a minimum involvement with the technical computer will liberate students to use it in the service of highly personalized design exploration. We saw that Professor Purcell believes that complex information systems force students to build models of how they work, increasing analytic and reflective capacities. But the models students build must be *personal.* And another way to describe Purcell's position that students should come to a "critical stance" about computers is that he advocates personal appropriation on a philosophical as well as technical level.

Thus, Architecture and Planning students work with a faculty which gives permission for a personal appropriation of the computer. And students do develop a ranges of styles. There are many dimensions on which to catalogue these styles. Here we discuss one among many, chosen both because of the frequency of its appearance and the significance of its educational implications, what we call planning versus bricolage. These can be seen as two styles of mastery; more accurately, though, they define a continuum of approach.

Bricolage

Bricoleurs approach a task, including tasks at the computer, as though they had just entered a foreign territory and they are there for the pleasure of its exploration. They work without a precise plan, often beginning with one element and "playing with it," allowing one idea to lead them to the next. They let their first idea change into something else or into nothing at all. They enjoy arriving at a method because it made sense after experimentation rather than because it was validated for some theoretical reason. For example, Cara, a graduate student in Architecture, contrasts traditional design studios where you are given a principle such as "design by aggregation" and the computer design studio where you can "do it" and "create your own theory."

Bricoleurs consider imprecision an essential element in design. Imprecision is ambiguous and ambiguity is evocative, stimulating new associations and insights. Bricoleurs frequently mention the importance of allowing themselves to be in such states and they use the computer as an ally for achieving them. For example, by keeping track of specific information, the machine takes a "bookkeeping" burden

off the designer who is then free to wander. The computer's speed and three-dimensional possibilities open the door to imagination.

Jim is an undergraduate in the Architecture program who uses Autocad (an electronic drafting system with limited three-dimensional capabilities) in his design work. Jim approaches design as an exercise

> where I randomly ... digitize, move, copy, erase the elements—columns, walls, and levels—without thinking of it as a building, but rather as a sculpture ..., and then take a fragment of it and work on it in more detail.

As he describes his style, he likens himself to Cara and describes both of them as "bottom up."

Like Jim, Cara also uses Autocad to design "abstractly, loosely, without looking at the details." Unlike many of her peers, she does all her rough, preliminary sketches on the computer. She uses it to try out options. She describes the process by which she worked on her class assignment to redesign MIT's new Arts, Media and Technology building in a way that makes it sound like "visual tinkering." General notions of "public space," "social activity," and "circulation," led her to the idea of "positive and negative space," and then to some first designs. This initial manipulation of simple elements led her to introduce "interlocking spaces" and "levels." All of this brought her back to thinking about how to use space to facilitate communication among groups within the building, and to another round of changes in her design. The process was circular; she found it pleasingly so.

Cara's path is open to associations; for her, "mistakes" are useful. They are stepping stones in a process wherein there is no premium on proceeding linearly. On the computer, Cara easily moves from the small to the large scale. The computer has created an environment where her preference for learning in this way can be appreciated as an asset. Cara's bricolage can be described as a conversation among herself, her design, and the computer. Her experience of the process is that each responds to the others. She uses the computer in the idea-generating stage of work; she accepts that it shapes her ideas. Indeed, Cara welcomes input from the machine. She feels she gets better results when the machine takes an active part in the design process.[7]

[7] Several student bricoleurs refer to the computer as a "sketchpad" and analogize its use in design to the way they use word processing. In writing as in sketching, their creative effort includes placing themselves in the role of passive observers of spontaneous developments. Mark, an undergraduate in Architecture, uses the word processor to "write anything that comes to mind" and then "discovers

Jim, too, makes it clear that openness to the computer's contribution is part of how he thinks about design.

> You ask the machine to do something for you. After it does it you analyze it, in your own way, distill the information out of its output, by, for example, sketching by hand on it, adding or taking out parts. Then you feed the changes back into the machine and see how it develops further ... and what else you can add or remove. .... If you keep going back and forth it is a much more rewarding experience.

Bricoleurs enjoy learning about the computer by working at the computer. They want to explore it on their own. In contrast, planners want to take the shortest distance between two points. They want to be taught. When they sit down at the computer they want to have already learned what they will need.

The Planner's Style of Mastery

Bricolage is characterized by the desire to explore and "tinker." In the planner's style of mastery, problem solving tends to be structured and progress is defined as moving along a path towards a predefined goal. Planners talk about the necessity for a universal "method." ("If you have a method, you can apply it to anything.") For example, Lloyd, a graduate student in Architecture, talks about always beginning from the "big picture."

> You come in with a design problem and with your preconceived ideas about what the solution could be like. You start generating what it is that you want, first diagramming it in large scale, creating the overall picture, developing the organizational concepts—which depend, for example, on the kind of atmosphere that you want your building to have. For example, for the redesigning of the AMT building, I had this idea in mind from the beginning. There would be a series of sculptural elements sitting in space with a pedestrian path going through and above parts of the building. .... This happens all the time, I never design randomly, I always have an idea generator for my design. .... With this larger idea in mind, you have a diagram which gives you direction, a framework for your decisions. .... Then you can start, form generating, sketching, shifting to small

---

the structure within what I have written." Debbie, a graduate student in Planning, describes her writing method in a similar way, "I do the free writing on the terminal if I have some ideas but I don't know what the structure is going to be like. .... I type out my ideas, let them sit for a day or two, and then come back and rearrange it into a paper."

> scale, embellishing, editing, transforming. .... Then you can zoom in onto the details, work on them to reinforce the bigger idea, which you always have to keep in mind.

Bricoleurs looked for input from the computer; planners want to dominate it. Lloyd is "willing to use the computer in the design process as I already know it. .... I use it within a precise design technique." Under no circumstance is control to be abdicated. Lloyd says that "it is easy to let the computer model the way you perceive things ... but then you have lost the game!" The machine needs to be kept in its place. Technical expertise is a way to protect yourself against its incursions. For the planner, lack of technical understanding can feel like being out of control.

Alan is a graduate student in Architecture who exhibits the planner's style of mastery. He gives no ground to the computer: "The computer doesn't change the way I design, because I first do the sketches on paper and then use the computer when I have the total idea of function and form, not before." He is usually quiet and calm when he works at the computer, but the exception illustrates what the rule is all about. "Once I tried to save a file [he explains in an agitated voice] and ... the computer went crazy and lost all the files! Sometimes it is the Athena people's fault, because they decide to change things and they don't tell you .... I was very angry and I did not want to work with it ever again." The most upsetting part of this experience was not the inconvenience of losing the files, but the "not-knowing-what-it-was."

While the bricoleur values the fact that the computer "pushes you to play," the planner sees this as threatening. Roger fears "getting lost" in computers. When this happens the machine has become "not a tool but a toy. Playing with the machine can be fun, but it is not architecture." For the bricoleur, machine-play feels like a creative part of the design process. For the planner, process should be a means to an end, and preferably, the shortest and most efficient means. Time spent figuring things out on your own is time wasted if straightforward teaching would have taken less time. Roger makes this clear: "Learning by yourself is not the best way to learn," and Alan had a lot to say about what would be his favorite way of learning about computers. It is *not* the way things are currently done.

> Form Processing should be set up in a two-part sequence where you learn how to use the computer first and then you learn to use it as a design tool. .... In general, one does learn by use, but learning design by use seems awkward .... There should be more hands-on sessions with the professor in the lab, to explain

> things directly, because for many procedures you waste a lot of time going around the problem rather than tackling it.

Bricoleurs are usually content to use software as it is presented to them, and manipulate it to fit their creative purposes; planners usually crave a more transparent understanding of what is going on. So for example, while bricoleurs see programming as an option, planners tend to see it as a necessity for effective control, "the only way to use computer graphics in an all-out way." [8]

In general, Architecture and Planning students would like to have a clearer idea about the computer's role in their future, but for planners it seems particularly important. They tend to construct a firm notion of where the computer will fit. Lloyd, for example, has given this a great deal of thought. He is certain that computer literacy means programming, but wants to postpone learning it until he can control his involvement:

> I want to wait ... because I am so ordered and structured that I could get wrapped up in it really easily. .... My idea of where I want to go with my career does not include sidetracking into the development of computer software.

Taking Account of Styles

Cara, the paradigmatic bricoleur, feels little tension between her style and the computer. But other students do feel a discrepancy between their way and what they perceive as the "computer's way." This is true for both the planners and the bricoleurs. Some planners find that, to their surprise, the computer (and the way their professors use it) pushes them towards a too-experimental style: Lloyd says that this is like a kind of painful therapy: "If you haven't quite crossed this barrier to thinking abstractly, loosely, frivolously and unorderly, then the computer might help you." But despite Lloyd's characterization of a computer that can help you to "loosen up," Autocad is a relatively rigid system. Bricoleurs adapt it to "their" way, but some feel distress. They worry that they do not think the "right way," and they feel a tension between their style and what Professor Miller described as the computer's "set of ferocious constraints." For example, Marcia fears the rigidity of the system. Working with the computer demands constant vigilance:

[8] Some bricoleurs even saw programming as potentially limiting. Thomas, for example, felt that the investment you made in it could make you less willing to deviate from your plans and you "learn from deviations." A little programming knowledge could be a dangerous thing.

> If you just look at the fine, perfect, neat and clean sketches that come out of the machine and if you don't change the output of a program, look at it, think about it, color it, keep going back and forth, you might forget that there are other ways of connecting the space or other things that can be done. .... You might end up mesmerized by the hard-line quality and definition of the output and not want to change anything. And even if you change, you do it in chunks, rather than making little moves, fine tuning, the kind of thing that comes from overlaying the tracing paper on an existing drawing and working on it. Everything you do is a very definite "thing." You can't partially erase things, smudge them, make them imprecise.

A soft, exploratory style demands access to a flexible tool. The computer in its more rigid embodiments, like Autocad and other drafting packages, is often too cumbersome to be helpful.

> With Autocad many things are not possible or take extremely long to achieve. With a real three-dimensional model it is much easier. .... You see that something might look nice, you cut a piece of cardboard and put it there. With the computer you have to think about how to put it there and if you can. And after a certain level of sophistication and complexity of the drawing, it becomes more difficult because you need specific commands for determining the height, size, or location of the piece you want to add.

Problems arise when educational computing presents itself as blind to questions of personal style. It leads to frustration and a sense of mismatch between people and the material, the feeling we have noted of people feeling that they don't think "the right way." It sidesteps the need to develop systems that will facilitate the work of people with different styles of mastery. And it sidesteps the need to rethink pedagogy. The combination of Professor Miller's "soft" presentation and the rigidity of Autocad led to the planners being confused and the bricoleurs being frustrated. There are no simple solutions, there is no absolute "right way." But it is naive to launch large scale experiments in educational computing that expect diversity in content but not in the form and the feeling of computer use.

Coda: On Technocentrism

MIT is at the cutting edge of technological advance and tends to measure achievement in relation to that cutting edge. Other universities found ways to provide students with "off the shelf" personal computers and made them the centerpiece of their educational computing programs. But in consonance with its technological values, the MIT Athena project made an early decision that state-of-the-art hardware would be its centerpiece. So, for example, the decision to group Athena computers into clusters rather than distribute them to students was made in the anticipation that the project would take its ultimate shape around advanced workstations.

In practice, the most advanced equipment was long delayed in its delivery. The full impact of the Athena program as originally conceived has yet to be experienced, but the computer presence at MIT has been strong enough to gauge the impact of the technology-centered Athena philosophy on the growth of computer cultures within departments. In Architecture and Planning we see its impact on two departments whose intellectual traditions are only partially rooted in the technical world.

In this setting, students have high expectations of the computer, reflecting the faculty discourse about a visionary future. In the present, students want and expect more from computers today than they are able to provide. When students' expectations are not met, they are disappointed. But, somewhat paradoxically, the disappointment is in themselves rather than in the computer. Students end up with a sense of their limitations, rather than those of the machine.

There is also evidence that the "technocentrism" of the Athena program has encouraged the development of a two class system. The two classes are not computer users and non-users. Rather, one class uses computers in a state-of-the-art way and a second group uses them in ways that are not considered to be on the cutting edge. The latter group is frequently overlooked, or at least under-rated by those involved in state-of-the-art activities. Members of the second group come to feel that their computer use is "mundane," and that they are "lesser" users. The way they use computers may indeed be "lesser" from the point of view of the exploitation of advanced hardware. But the exploitation of state-of-the-art hardware does not necessarily define a successful educational experience.

Professor Karen Polenske spoke eloquently on this question. She says that "Athena has been good" to her. She has received support, developed new courses, and become involved in exciting areas of research. But her concern is real.

> To me, what's developing now and what bothers me, is that I think we are getting into very much of a class system. MIT has always had a class system of some sort, but Project Athena is making it much worse. .... In terms of students who get helped, or even faculty, if you're doing something right now on the RTs—something that is considered "sophisticated" types of computing—that is great.[9] You are patted on the back and given support. But, if you're trying to use what are considered to be just mundane, everyday things that they've already worked with for three years, or if you have a question that they think "anybody" should know the answer to, you may wait three weeks for an answer.

As Professor Polenske sees it, the "Athena people" have made it clear that work on the IBM RT is what Athena is supposed to be about. But, in her field, Polenske says there is still a great deal of work to be done with the ATs and Lotus, and she feels she should have the support to do it.

> Why should I be looked down upon if I say I want to develop additional programming capabilities for ATs? We haven't even touched one tenth of what the ATs are capable of, and already we're going on to RTs and fancier ways of doing things. Those of us who still want to work on ATs are made to feel that we're just not with it. .... Students working on Microcad (the newest graphics software in the Architecture Department) are considered to be better than students on Lotus because Lotus is considered to be old hat.
>
> I don't want to be forced by Athena—if I want their money—to spend all my time with new applications and mainframe work. That's not where I feel I can be most productive. I shouldn't be made to feel badly if I want to do something with existing software.
>
> Learning to use what exists and to teach it better are important parts of the learning process. Why is it not considered progress if I'm still doing the same thing I was doing the first year of Athena and trying to do it better? Do we always have to be using the latest equipment?

Polenske points to other consequences of the Athena emphasis on high-tech as prestige use: the emphasis on the machine tends to diminish the emphasis on training people who are essential for the education of students, the support staff and teaching assistants. Similarly, not enough attention is paid to supporting the realities of classroom teaching. For example, in the School of Architecture and

[9] The IBM RT is the most advanced system that IBM has contributed to Athena.

Planning there is no computer-equipped classroom that can accommodate more than twenty students, although many classes have more than twenty students enrolled.

The emphasis on state-of-the-art technology led Athena to adopt UNIX as a universal operating system. But UNIX programming can be a stumbling block for students without strong mathematical backgrounds. And UNIX does not support programs such as Lotus which are widely used in the real professional world as well as in non-Athena courseware within MIT. Athena policy tends to put students and faculty in a position where they use UNIX at "school" and DOS at "work." Athena has judged Lotus unimportant, but that judgment does not take into account the effectiveness of teaching with Lotus or the utility of Lotus experience when students look for their first job. It only reflects that Lotus software, like DOS, is not on the "cutting edge."

The "cutting edge" philosophy is, of course, most apparent in the "negotiated peace" that Athena has made with word processing applications. Seventy percent of Athena use at MIT is for word processing. Yet, word processing is not an "official" Athena activity. In the CRL, Athena did not put word processing on the system because, according to Professor Polenske, they "weren't supporting word processing. You couldn't use the core space for that." It was the School itself that funded the installation of word processing in the CRL. For many, it was disturbing to have to "fight" for a necessity. For Polenske, as for many faculty throughout MIT, word processing has become an essential educational technology. It means that students can be asked to rework papers. Polenske sums up her popular position: "Teaching and learning is writing."

> Investing in word processing is probably one of the best things we could do if they really want to get computers into the teaching environment. But that's something that doesn't come out in my Athena proposal—the way in which students are learning to become better writers. There's no support for that kind of learning in Athena. You'd be laughed at if you wrote it into your Athena proposal as part of what you were using their funds and equipment to achieve.

The decision to make Athena a cutting edge technical experiment, like the decision to make Athena a faculty-driven activity, has roots in MIT's traditions of innovation. These traditions have their strengths. They also have their costs.

Implicit in Athena is the idea that worthwhile projects will rise to the surface because faculty will find them worth continuing. But the technical emphasis of Athena has not encouraged the development of the educational criteria on which

to base these decisions. In addition, as we have seen, faculty do not necessarily see evaluation as their job. Even in an environment which encourages faculty initiative as the motor for innovation, there is a need for greater articulation of educational rather than technical aims. And even in an environment that is experimental, there is a need to take stock. Greater stress on the educational rather than the technical would put the faculty directly in that process.

Technical emphasis encouraged faculty in a belief that working with computers could be something of a "left handed exercise." The Architecture and Planning faculty went into Athena with enthusiasm and rich ideas about using technology for an enhanced learning experience. But making educational technology work is a complex business. In Architecture and Planning, we see faculty getting started in a long process. That process is not simply one of debugging software or of getting the right hardware. It is far more complex and involves, as only one example, a deeper appreciation of styles of designing, learning, and appropriation of technology. Faculty need to be supported in a program of study, conversation, and reflection that takes account of such complex issues. A project that assumes that "everything will follow" from the right choice of technology cannot provide this support.

Project Athena at MIT: Computing in the Department of Civil Engineering

by Wim Overmeer and Donald Schön

May 1988

Introduction

Method of Study

Between April 1986 and September 1987, the Project Athena Study Group team conducted four case studies of Athena at MIT. This report summarizes one such study in the Civil Engineering Department. Its focus is on one particular Athena-supported project, the Computer Aided Teaching System (CATS), undertaken by the Constructed Facilities Division. This report is based on information gathered through interviews with undergraduate students, graduate assistants, and faculty members in Civil Engineering. In a few cases, the researcher's participation in informal discussions among faculty members, or between Civil Engineering faculty and faculty from another university, provided useful information, as did some limited observations of students and faculty in classes and laboratories.

Between November 1986 and April 1987, we interviewed thirty people whose accounts are reflected in this report. Most were interviewed once, and a few of the central faculty actors were interviewed twice.[1]

We interviewed the three graduate students who had served as teaching assistants for those subjects in which Athena-sponsored programs had been tested. And we interviewed three graduate student research assistants who had worked with Professors Connor and Slater in the development of this software. In addition, we spoke with eleven undergraduate students who had used such software in courses and laboratories.[2]

[1] We began by interviewing David Marks, Department Head; then Professors Robert Logcher, Jerome Connor, and John Slater, the principal contributors to the Civil Engineering proposal. We interviewed all faculty members who had used CATS software in their courses: Professors Robert Whitman, Oral Buyukozturk, Eduardo Kausel, and Lorna Gibson, all members of the Constructed Facilities Division. And we interviewed all the faculty members we could discover who were known to have strong points of view about computers in education. These included, in addition to those listed above, Professors Keith Stolzenbach, Sue MacNeill, Yossi Sheffi, and Herbert Einstein.

[2] All of these students were members of the Constructed Facilities Division, although some of them were involved in more than one sub-specialty in Civil Engineering. Of the eleven students, ten were male; despite repeated requests, the researcher could not get more female volunteers to participate. Ten of the eleven students were in either their senior or junior year; one was a sophomore.

While the sample of undergraduates interviewed is small in absolute terms, it is significant in relation to the number of students currently enrolled in Civil Engineering, and it is even more significant in relation to the number of students enrolled in the Constructed Facilities Division. In the case of some subjects, we were able to interview most of the students who were enrolled when Athena-supported software was used.

We do not take our sample of students to be representative of students in Civil Engineering in the strict sense, but it is large and varied enough to be significant. We shall describe the patterns we found in our interviews, suggesting that they may well hold true for the current universe of Civil Engineering students—meriting, in some cases, more systematic study. Our interviews ranged in length from one hour to three hours. The interviewer worked from a set of guideline questions, but the interviews were open-ended to allow the exploration of unanticipated topics or issues of particular concern to the informant. All interviews were tape-recorded and transcribed.

Undergraduate students were promised anonymity, and their names have been suppressed or fictionalized. Faculty and graduate students were told that when specific attributions were made, they would have an opportunity to review our written account of their statements.

This introductory section presents a picture of the Civil Engineering Department when Athena began and a brief history of the Athena project within the department. It is followed by three major sections, corresponding to the major themes of our investigation:

- the actors and their intentions;
- student and faculty experience in relation to these intentions; and
- the educational implications of the match and mismatch between intentions and expectations, the impact of Athena on teaching and research within the Civil Engineering Department, and the larger set of educational and institutional lessons that we have drawn from our findings.

The Departmental Context: a Brief History of Athena in Civil Engineering

* Civil Engineering before Athena

In order to understand what Project Athena meant to the Civil Engineering Department, it is necessary to grasp the department's situation—as its faculty

saw it—in the early 1980s. The department was seeking to position itself anew in relation to the changing environment of engineering practice and the changing competitive environment of MIT. In this process, computers played a central role, both through advances in computer science and shifting views of their potential uses in engineering.

The entire field of Civil Engineering, the first engineering curriculum at MIT, has undergone a series of major transformations since the end of the Second World War. Structural engineering, long at the center of the field, was an "elevated craft" until that time. But after the war, the theory of structural behavior came to be seen as a special case of Newtonian physics, mathematically identical to the theory of electric circuit design, and by the end of the 1950s, advanced mathematics were intrinsic to structural analysis. Problems of structural behavior were represented by complex, multiple differential equations; and methods of numerical analysis, like numerical iteration, were used to solve these equations. With the introduction of digital computers, programs were written to automate equation-solving, and Civil Engineering became the first MIT department outside of Electrical Engineering to use them. During the second half of the 1960s, computational models for Civil Engineering used large mainframe computers such as the IBM 360. By 1970, STRUDL, a computer program based on the finite element method, widely used in the analysis of large-scale engineering projects, found its way into the classroom where it affected the education of a whole generation of students.

In the late 1960s and early 1970s, opposition to the environmental, social, and political implications of large-scale engineering projects led the civil engineering profession to a heightened sensitivity to non-technical consequences of its practice. The MIT Civil Engineering Department responded by hiring representatives of other disciplines—economists, institutional analysts, and cultural anthropologists—in the hope that they would contribute to a critical understanding of the non-technical assumptions underlying engineering models.

By the early 1980s, Civil Engineering was hit by a challenge of a very different kind—technical rather than social. The field no longer seemed on the cutting edge. Within MIT, undergraduate enrollment in Civil Engineering declined dramatically, while enrollment in Electrical Engineering ballooned to more than a third of the entire undergraduate student body. In the field of engineering as a whole, the status of civil engineering seemed to have slipped. As one faculty member in the department put it, "Every time a space shuttle goes up, our enrollment goes down!" Under this pressure, the multidisciplinary research and curriculum introduced in the 1970s came under fire from within the department, as

the department found itself caught in the financial crunch that gripped MIT as a whole. Tuition rose significantly, discretionary funds dried up, and competition for external research funds became more intense.

Meanwhile, personal computers began to be a presence in the department—some bought with research funds, others owned by students and faculty. And from the fields of Artificial Intelligence and Computer Science—the very fields that were so successfully drawing students away from more traditional engineering specialties—radically new approaches to computational knowledge were making themselves felt.

In this context, heated and turbulent, the department sought to redefine itself. In particular, the multi-disciplinary faculty departed, and the core group of civil engineers began to invest in new forms of computer-based activity.

Professor Jerome Connor, who had been one of the most active promoters of STRUDL, now championed the idea of the use of the computer for "knowledge engineering," and created GEPSE, a computer-based knowledge engineering environment. As Professor David Marks became department head, research and teaching were reshaped to take account of what Connor called "the AI Revolution." An AI laboratory was created in the department, in which both Professor Connor and Professor Robert Logcher took an active interest. In addition, there was a laboratory for applications of robotics in construction, and the department began a new undergraduate program in Engineering Systems and Computation under Professor Yossi Sheffi.

As an alternative to Electrical Engineering's introductory computer course, 6.001, Civil Engineering initiated its own introductory computer course, 1.00—a course intended, as one faculty member put it, not to teach computer science but to program a computer to solve an engineering problem. This course, it was hoped, would socialize students into civil engineering as a new, computer-involved discipline.

* How Project Athena evolved in Civil Engineering

Project Athena, begun in May 1983, presented itself as an opportunity to a Civil Engineering Department where the computer was already central to a process of active self-transformation. Athena represented a new source of computational resources, and for some faculty its goal of establishing a coherent network offered an appealing alternative to the fragmented clusters of personal computers that had already made their appearance. And Athena presented itself as an

opportunity to fund educational "courseware" that certain faculty planned to develop as a part of their strategy of computer-based transformation. They thought of themselves as representing a "new wave."

Professor Logcher, a strong advocate of computer-based learning within Civil Engineering, had been involved in the discussions that led to the formulation of Athena's goals. There was, as he put it,

> ... a feeling that the educational process hadn't changed in centuries. [It was] basically a labor-intensive process that required an instructor to present material, sense the [student's] reaction and go through a process of understanding the reaction of the students to the presentation of materials, and then judge whether or not the teaching was being effectively learned.

In Logcher's view, Athena had an educational hypothesis, and it was that "computers could make education ... more efficient and ... effective."

> We felt that we'd like to prove that at least parts of that system could be taken over by an intelligent computer system, maybe not replace the instructor, but make him more effective.

For Logcher, Athena's main challenge was to develop educational "courseware" which needed to be in place "before we could actually go to the point of asking each student to come in and pay for a computer and use it effectively in his education."

As things turned out, Project Athena took root in Civil Engineering in only one of its three main sections, the Constructed Facilities Division, which put together a long-term proposal covering five years of work, with $112,000 requested for the first year. The seven faculty members involved in this proposal saw computers as potentially revolutionary for structural analysis and design.

The proposal envisaged a shift from "traditional structural components" to computer-aided design of "engineering systems." Its proposed Computer Aided Teaching System (CATS) would be a unified intelligent tutoring system for both graduate and undergraduate education across the curriculum. In addition, the CATS proposal hoped to allow "quick implementation of a new graduate program in Computer Aided Structural Design." Prominent among the seven faculty members involved in the CATS proposal were Professors Logcher and Connor, longtime champions of computers in education, for whom the new proposal was an opportunity to "redo the sixties" when both had been involved in an earlier round of computerization. Both men were prepared to make a career shift—Logcher, from construction management to structural engineering; Connor, from expertise in boundary and finite element methods to "knowledge engineering."

Project Athena made a very limited response to the CED proposal: it allocated $35,000 for the first year's work. For the rest, it would "wait and see."

Of the seven faculty members involved in the proposal, all but two—Connor and Slater—withdrew. Logcher, when asked to explain his reasons for withdrawing, mentioned that Athena's response confirmed his earlier hunch: it would take Athena much more time to acquire the research funds necessary for the full project. Moreover, Athena had insisted on hardware and software incompatible with the teaching games Logcher had already developed. Among several of the faculty members that had been involved in the proposal, there was the feeling that Athena's initial response showed that it was "not serious about education." They felt there was not enough funding over a long enough period of time for them to make a commitment that would involve, as they saw it, a change of career direction. (Over four years, however, the CATS project did receive a few hundred thousand dollars in Athena support.)

Connor and Slater became, then, the principal actors in the Department's Athena-supported project—Connor, a senior professor long committed to computers in education; Slater, an assistant professor, educated as an undergraduate in the Department. The CATS project, to which we now turn, was marked by their interests and intellectual personalities.

Part I. Educational Intentions

In the mid-1960s, computational methods based on an *algorithmic* representation of knowledge had become, according to Connor and Slater, a "way of life" in structural engineering. By the early 1980s, advances in computer science had changed the thinking of those concerned with computers and engineering. In particular, Artificial Intelligence research had developed *rule-* and *knowledge-based* methods of representing knowledge. Connor and Slater believed that rule-based representations of knowledge allowed a new way of looking at engineering. The focus would shift from one-time, detailed design to iterative preliminary design where large chunks of detailed design could be automated through the use of algorithms, and preliminary design could be supported by rule- and knowledge-based design advisors. The central idea was to capture the knowledge of professional structural engineers in "expert systems." In their writing on the subject, Connor and Slater were optimistic:

> The ability of the design engineer to tap this [heuristic, experience-based] knowledge at the computer terminal when he or she is working on actual design of a system, will be of immense help. On the other hand, the interaction of an expert with a computer should help the computer itself in gaining experience over and above the stored knowledge .... Design codes usually contain rules that also are non-algorithmic or heuristic in nature, and which in a sense embody the collective wisdom or expert knowledge of the profession.

Their optimism about expert systems extended to education. Along with the development of a computational design advisor, Connor and Slater envisioned an intelligent tutor for undergraduate engineering education. It would be able to detect and respond to conceptual weaknesses in students' understandings of subject-matter. Connor and Slater divided "classical" engineering education into three fundamental modes: 1) "being told" concepts and techniques, 2) "addressing their applications," and 3) "doing-using" and being corrected. Each had a kind of teaching most appropriate to it: "being told" used the tutorial, "applications" used homework and problem sets, and "doing/using" relied on laboratory and design projects.

Connor and Slater believed that each of these modes had problems that the computer could solve. In tutorials, teachers had a hard time grasping whether students understood fundamental principles. In problem sets, students were either caught up in massive amounts of tedious calculation, or were condemned to plow through trivial examples. In design/laboratory projects, students could write "conceptual papers" but had no opportunity to "write computer programs and make them work." In all three cases, students were confined to "passive learning." Connor and Slater wanted to use the computer for "interactive learning." First, the computer-based tutor would diagnose and correct a student's "causal reasoning." Second, as a design assistant, the computer would take on the tedious work of analytic calculation and would free the student to conduct multiple iterations of design and analysis. Third, computerized data-base management would accumulate "design knowledge" which students could then bring to bear on their design projects. With all this, students would be able to address larger, more complex, and therefore "more realistic," design problems.

Connor and Slater focused their efforts on the development of their knowledge-based advisor and tutor. Although one might have seen this computer-oriented research project as requiring a basis in a study of engineering practice, Professor Logcher gave voice to the assumption shared by all involved faculty that there seemed little need to begin with such a study:

> We just wanted to get on with the development of the tools, and we were convinced that the group hypothesis would be so obvious that no one would question it.

Three specific software development projects were proposed—each an interactive version of a traditional teaching format:

1. an intelligent tutor to complement the teaching of elementary structural mechanics or the statics of determinate structures—one for beams, MACAVITY, and one for trusses and frames, FATCAT;[3]
2. a program to assist in designing indeterminate structures by carrying out their analysis, GROWLTIGER; and
3. a design advisory package, General Engineering Problem Solving Environment, or GEPSE, to support students in the preliminary design of structures.

MACAVITY

MACAVITY, intended for all sophomores who have chosen to major in Civil Engineering but have not yet chosen a specialization within the field, is a tutoring program designed to help a student find his or her way in the world of determinate beams.

Beams are, conceptually, the simplest type of structure. They are considered as building blocks for more complex structures and as elements to which, under special conditions, some types of complex structures can be reduced for the sake of preliminary analysis and design. Because of their simplicity, beams are used to teach students: 1) how to represent physical structures; 2) how to understand the meanings of fundamental ideas such as load, support, force, shear and moment; 3) how to understand the relationships among these ideas; and 4) how to represent and simulate the behavior of structures by means of a set of relatively simple mathematical equations.[4]

---

[3] FATCAT has not yet reached the state of development where it can be tested and so this report will not discuss it further.

[4] In the case of a determinate structure, one might think of a ruler supported by two pencils. The play of forces and the physical behavior of the structure can be calculated without taking into account how the beam deflects as a result of the loads. This simplifies the problem, since equations can now be formulated in terms of force alone. In the case of an indeterminate structure, for example, a ruler fixed at each of its two ends between heavy books, the deflections resulting from the loads need to be taken into account and expressed in a series of equations. This set of equations must then be related to equations that describe the play of forces, using assumptions about the relationship between forces and deflections, according to Hook's law.

Well before Project Athena, Professor Logcher had proposed the idea of an intelligent, computer-based tutor, advancing several of the ideas that were later incorporated in MACAVITY. Logcher wanted to go beyond "multiple choice" tutors, like PLATO, which focus on drill and practice.

Logcher based his thinking on a program called SOPHIE, a knowledge-based diagnostic program for electric circuit design. He proposed to develop an intelligent tutor that would assist a professor of structural engineering in the classroom. He suggested that a professor's intuition *and* judgment could be captured in a knowledge-based system which could then be used to relieve the professor of a part of the time-consuming task of coaching students.

When students are given problems designed to help them become familiar with fundamental concepts and methods, Logcher observed, they make mistakes. Sometimes, these result from plain oversight and sometimes, from "conceptual weakness." In order to help a student, the teacher must determine what *kind* of mistake the student has made, which requires him to go step by step through the student's reasoning—only to discover, perhaps, that the student has made a simple miscalculation.

Slater's MACAVITY program took over the idea that an intelligent tutor for the analysis of determinate structures would assume the labor-intensive process of coaching students in fundamental concepts and methods. In addition, Slater put forward several other ideas.

First of all, Slater emphasized the graphic representation of problems, developing a graphics system in which students could draw structures and visually inspect their deflections under load.

Second, he had noticed that students who present "blank faces" in class, the ones most in need of help, are likely to be too embarrassed to ask questions, especially in the presence of their peers. As a result, a teacher has to wait for exams—usually half way through the semester, when it is perhaps too late—to become aware of the most serious problems. The longer students wait to ask questions, the further they fall behind, the more embarrassed they become, the less likely they are to ask questions.

Slater designed MACAVITY to attack this vicious circle of conceptual weakness and embarrassment. He proposed that students could use it with *anonymity.* MACAVITY would allow students to ask as many questions as they wanted, as often as they wanted, without having to suffer the embarrassment of making public how far behind they were.

Finally, Slater intended to *control the student's representation of the problem.*

In essence, MACAVITY solves mathematical equations that describe the behavior of any determinate beam when a given load is applied to it. The student may ask the program to present such a problem, or he or she may invent a new one. The student types in the mathematical equations describing the forces at work on the beam in order to calculate the force the beam exerts on its support. The program monitors the student's problem-solving process and interrupts when there is a mistake.

Slater describes the program working in three modes. There is an "expert" mode in which it solves the problem, "emulating a professor in the classroom working on a blackboard." There is a "question-answering" mode, in which the student can ask questions in the form of, "What if I tried this part of the equation?" And finally, there is an active, "learning by doing" mode, in which the student solves a given problem, the computer compares "what is supposed to be there ... to what is actually there," and gives an explanation of the discrepancy.

As long as the student does not make a mistake, the computer does not interrupt. But when there is a mistake, the program follows a strategy of *successive hinting*, giving increasingly specific hints until the student produces the correct answer. The student may then ask to test his or her understanding with another example or request a grade for the complete exercise.

Interruptions are aimed at correcting weaknesses in what Slater refers to as the student's "causal reasoning." By this, he means the ability to break a problem down into the equations and force diagrams that apply to it, to understand the relationships among these equations and diagrams, and to find a solution through a chain of if-then propositions that link states of equations and force diagrams to strategies of solution.

Slater makes what he considers a crucial distinction in problem solving between causal reasoning and "mechanical pattern matching." The latter, he illustrates as follows:

> ... you give an example in a lecture, and then in an examination ... present the identical question [with the] figure rotated about two axes so it's turned backwards. And the way some students solve it is that they figure it out and *they turn it back to the thing they've already seen before.* It's clear that what they're doing is repeating a mechanical task, but they don't exactly know how it is that they're supposed to solve it.

In order to counteract mechanical pattern-matching, MACAVITY forces the student to use a mathematical representation of the problem. It stands in the way of a solution through the "mechanical" manipulation of an image. The intention

is to make explicit a student's causal reasoning so as to be more able to correct it when faulty.

In MACAVITY, then, Slater has incorporated several educational hypotheses. The computer will reduce labor-intensive diagnosis and coaching, graphics will enhance intuitions of structural behavior, anonymity will counter the embarrassment that keeps some students from asking questions, and control of the student's representation of the problem will improve causal reasoning.

Slater and his graduate assistants now call several of these assumptions into question. They ask whether professors should be able to monitor a student's use of the tutor when they feel the student is falling behind. They wonder whether the inclusion of menus, that reveal options for problem solving, may inappropriately reduce the student's need for independent thinking. They debate strategies for "hinting" and for fine-tuning the program to the right level of difficulty.

GROWLTIGER

The development of GROWLTIGER is rooted in Slater's criticisms of an earlier generation of computer design assistants for structural engineers (in particular, the STRUDL program that was in use in the 1970s when he was a student at MIT). STRUDL incorporated finite element and boundary element methods that Slater as "exotic math for general purposes,"

> ... like cranking out a machine; every problem became reduced to the finite element method, and then you pulled the lever.

But with these procedures, characterized by Slater as "a pile of numbers," "you don't see anything." Students who learned structural engineering in this way "could do linear algebra" but "damned few know how to detail a beam and put together a structure and make everything work."

When they introduced the finite element "crank," professors gave up the old-fashioned, graphic methods that "showed you how things worked." These were less accurate, but "they required you to think interactively about solutions," rather than merely to "design which crank to use." They were "problem-oriented," and demanded "physical intuition;" "you needed to do it a zillion times."

In addition to these graphic methods, there were approximate methods where you

> draw what you think the thing is more or less going to look like, and then use that to get answers in terms of forces, moments, shears, etc.

With the older methods, a problem in moment distribution would be solved by holding the construction fixed, relaxing local members one by one, and then seeing what happens to the whole. The picture of the entire structure would be achieved by superimposing the effects of each member on all other members, and of all other members on each. Slater believes that these methods, interactive and visual, kept students involved, gave them a better sense of how forces on a structure create local forces in members throughout it, and how deflections in the whole structure are a result of the interplay of forces and structural characteristics. Nevertheless, the older methods had problems. It was difficult for students to superimpose the effects of various members. And even one iteration of design and analysis required a great deal of number-crunching, all of it vulnerable to error. So Slater set out to develop software that would recapture what had been lost in the sixties—the interactive graphic analysis of the old-fashioned methods—while retaining the analytic power of finite-element analysis. The result was GROWLTIGER, a program based on the same mathematical representation used in STRUDL, matrix algebra. But unlike STRUDL, GROWLTIGER is both graphic and interactive.

With the version of GROWLTIGER in use by the first semester of 1986, a student can either build a structure of her own design (as long as it does not contain curves) or select one of the four types of structures displayed on the screen, and modify it. A structure like a truss or a frame is represented on the screen as a pattern of lines, where lines represent members and their intersections represent nodes. There are a few types of members and a few types of nodes.

The student need only set the geometry of the structure, select the dimensions of the members, the type of steel to be used, and the loading system. He or she can then ask the computer to analyze the forces, moments and deflections, and the program will display them graphically on the screen. In order to provide a better feel for the impact of the loading, the stiffness of the structure and the resulting deflections, the program can display the deflection moving from zero to the full range.

In designing indeterminate structures, the designer's predicament is that he can't design without an analysis, and can't produce an analysis without a design. To deal with this issue, GROWLTIGER has a default setting that reflects common design practice with standard-sized beams and standard material. After one

iteration of design and analysis, students can check whether their structures behave within the range of allowable tension in the beams and allowable deflection of the structure. They can then decide to change the geometry, the loading, or the stiffness. They can redo the analysis in a matter of seconds and save any or all the structures designed and analyzed. Using various criteria—for example, minimal number of members or minimal weight—students can optimize their designs.

Slater intends GROWLTIGER to be used as a *design tool* that relieves the tedious number-crunching work of structural analysis, and as a *virtual laboratory* for experimentation in structural design. A student will be able to create a graphic representation of a structure, make a local modification in it, and then explore the consequences of his move for the behavior of the structure as a whole.

In this way, Slater hopes that GROWLTIGER will help students gain an intuitive grasp of structural behavior. Because it short-cuts tedious equation-solving, it allows the student to focus on "the more conceptually difficult portions ... describing the geometry, and shaping the structure, and really engineer the structure to fit the purpose." GROWLTIGER will enable the student to "do much more interesting designs" and "really [experience] what real-world engineering is like."

Finally, GROWLTIGER will allow instructors to teach in in such a way as to leave students free to make critically important discoveries about structures for themselves:

> What you can do is ... just tell the student, there are these five or ten different kinds of ... structural systems that are classical. Go experiment with them, and find out what it is that makes one different from the other. Come back and tell me. And you guide what they do. They will try the right things and make the right discoveries for themselves, but the distinction is that they are *discovering by observing results of these computer models—how the things actually work*—which is enabling them to do what many hours of work would have been required to do ....

GEPSE

GEPSE, the General Engineering Problem Solving Environment developed by Professor Connor and his graduate students, is a set of computer programs that combines an AI shell, an operating system, a data-base management system and a knowledge-base system. Together, these make up a computer environment in

which a student can write specific engineering-oriented computer programs of his own, without having to either run canned software or master AI thinking.

For Connor, GEPSE is less a particular piece of educational software than a "dream" that will become "a way of life around here." For him, programs like MACAVITY, FATCAT, and GROWLTIGER are elementary versions of the sorts of program that will ultimately be developed in a computer environment like GEPSE.

Unlike Logcher, who envisioned expert systems that could emulate professors, Connor envisions a computer revolution in engineering practice. In the aftermath of the AI revolution, he believes, graduates who cannot develop computer programs will be unable to find a job, or will become routine engineers doomed to replacement by a computer. Connor believes that students need to prepare for a practice in which the computer is a way of life, and need, therefore, to develop their own programs—adding to existing knowledge- and data-based systems or creating new ones.

With a fully developed GEPSE system, Connor believes that the computer will become a "laboratory facility for knowledge-engineering" where students will "play around with knowledge," mixing knowledge components much as a chemistry student mixes chemicals, to "see what happens."

For Connor, the new computer environment will allow students to become "developers, innovators." They will take the place of courses like "reinforced concrete beam design" that turned out "little robots," technicians "finely tuned to solve specific problems and obsolete as soon as they left the gate."

In the course of the debate within the Civil Engineering Department about the idea of GEPSE, Connor argued that the department should, in the interest of academic and professional autonomy, develop its own AI shell. He claimed that "canned shells," available on the market at the time, were too expensive and ill-suited to engineering purposes. So he proposed to develop an AI shell within the department, programmed in C, and taking advantage of Athena hardware. The program would combine high-level AI ideas with lower-level engineering knowledge, integrating algorithmic and rule-based programs.

In the division of labor Connor imagines, undergraduate students will develop the computer programs. In particular, he sees freshmen and sophomores as "wizards" who "run circles around the older kids" and "love to write software." Older students, more mature, are "more interested in using the programs to develop specific knowledge-based systems." And faculty are "way behind, out of this." They will have to climb on board as best they can, once the department has

been "turned around." Indeed, Connor sees himself as engaged in a "crusade" for his vision of the role of computers in the department—a crusade legitimized by Project Athena, by other research in the department, and by the department's visiting committee. He is fighting against "die-hards" who believe computer work does not have intellectual content and resist investing in the development of an AI shell specific to Civil Engineering; against those who want "super equation-solving programs with super computers," "pure algorithmic stuff;" and against the use of pre-packaged rule-based AI programs.

The current state of GEPSE's use in the department falls considerably short of Connor's ultimate hopes. At the undergraduate level, only one subject uses GEPSE. In his own graduate courses, he uses the program as he would like to see others use it. Instead of merely writing a term paper, his students work on a real-world problem, using programs they write on GEPSE.

> They've got to research the problem, define the issues, identify how it is solved, establish priorities, put it all together, and make the damn thing work.

Professor Herbert Einstein is currently using GEPSE to develop a "geotechnical tutor." It will consist of three modules. The first is a physical geology tutor. In a second, students apply their geological knowledge to case studies in which they are asked to plan explorations of a site, for construction purposes, on the basis of photographic images and information about the site. Eventually, Einstein hopes that each student will develop his own geological rule- and knowledge-base. In a third module, students will be asked to select a foundation system, under conditions of uncertainty about actual subsurface conditions, and then to compare costs of further exploration with changes in expected construction costs.

These applications will use recent innovations in computer graphics. Lecture slides of site formations will be transferred to videodisc, so that students can view them on a terminal screen where they will be able to outline features of a formation instead of selecting from a menu of possibilities. (This is an adaptation of a method that had been developed with Athena support for projects in several departments, including one project on brain anatomy.) And the computer will be used to construct three-dimensional pictures of geological formations, based on profiles of selected borings—an application of Slater's three-dimensional image program for GROWLTIGER.

So far, however, development of these geotechnical programs has not advanced to the point where students can make significant use of them.

Nevertheless, Connor believes that Slater's programs are working well with students at the undergraduate level, almost all of whom he sees as "comfortable with computers." He "does not hear them scream," and "nobody knocks on his door." He believes that teaching assistants can handle what problems there are and that students working in the computer laboratories have good relationships among themselves.

Part II. The Programs as Experienced

Main Findings

1. The computer programs have taken on an existence of their own which is social as well as technical. This social existence has influenced how the programs are experienced. *It is not only the programs' design but the meanings attached to them by the various actors involved in their development and use that determine this experience.*

2. *The development process has been fragmented and disjointed.* Connor and Slater, the principal developers, have promoted the programs' use in the department. Teaching assistants have played important roles as advocates of the programs and have been closest to the students using them. But among faculty, there are sharply divergent views of the educational value of the CATS programs so far developed. On the whole, faculty members have been indifferent or negative to the programs. Those who have introduced the programs into their courses have, with a few important exceptions, tended to distance themselves from the development process. They have treated the programs as marginal additions to existing courses. They have not learned to use the programs. And they have not participated effectively in the programs' evaluation and further development.

3. *Faculty and student attitudes toward CATS have been influenced by what one faculty member called the experience of "scrambling to get the programs working".* The difficulties caused by bugs in Athena hardware and operating systems and CATS software have figured prominently in descriptions of experience with the programs on the part of both students and faculty—those negatively and those positively disposed toward the programs' educational potential. It is difficult to distinguish reactions to bugs from reactions to the programs' intrinsic value.

4. *In the very limited exposure a few students have so far had to MACAVITY, what comes through most clearly is the discrepancy between intention and outcome.*

Students have found it "childishly simple," and have reacted to this simplicity—and perhaps also to the constraints built into the program—by creatively "beating the system."

5. *Undergraduate students' exposure to CATS has been mainly to GROWLTIGER, and their reactions to it have been overwhelmingly positive.* Of the students we interviewed, most have found GROWLTIGER to be, as Slater hoped, an easily usable tool that relieves them of the burden of tedious calculation in structural analysis and design. They speak most glowingly of their use of the program in independent design projects, reflecting, in part, their thirst for design experience in a department they see as giving too little attention to design. Some students emphasize GROWLTIGER's effectiveness in helping them gain new understandings of structural behavior. Especially striking were instances in which students gained new insights into structural behavior as they puzzled over surprising results that the program had produced.

In contrast to these positive reports, there was also a counter-theme in reactions to GROWLTIGER: a concern that the program's automatic analysis can mask "what's really going on" in analytic work.

6. *There is a high degree of diversity in students' experience with GROWLTIGER. Different patterns of experience are clearly connected to students' experience with and attitudes towards computers and their views of Civil Engineering as a special field of study.* The majority of students in our sample who say they do not like to program computers and want to limit themselves to the *use* of existing programs, or even to *avoid* computers to the extent they can, also tend to give *instrumental* descriptions of GROWLTIGER; they see it mainly as a useful, time-saving analytic tool. And they tend, further, to describe the field of Civil Engineering in terms of its *opposition* to computer programming. Students who say they like to program computers, do a lot of it, and see themselves as good at it, tend to describe GROWLTIGER in more subjective terms that underscore its value as an experimental environment, a "microworld." These students describe the program not only as a useful analytic tool but as an environment for experiment, theory-testing, intuition-building, and playful exploration. These students do not describe their choice of Civil Engineering as an alternative to computer programming.

Students who express negative attitudes toward programming tend to see it as "abstract," "theoretical," "boring," or "miniscule," and contrast it with the "concrete," "practical," "interesting," "large scale," and "socially relevant" activities of Civil Engineering. These students often take an additional step, describing

themselves as *the sort of person* who abhors the former and embraces the latter.

7. *How students respond to their pivotal encounters with computers depends greatly on the ways in which they deal with the inevitable frustrations produced by "bugs"—whether they like to deal with them, can tolerate dealing with them, or find them overwhelmingly discouraging.* These reactions are linked to students' particular ways of seeking assistance.

8. *Finally, the students' ways of perceiving and dealing with bugs in the computer environment are linked, especially in freshman and sophomore years, with their perceptions of the larger MIT environment and their attempts to work out a viable adaptation to it.* Within this context, students' first exposures to computer programming in introductory subjects at MIT have a strong influence on their attitudes toward computers and help to set the course of their further academic careers.

Student and Faculty Experience with MACAVITY

This is a short story. At the time of this study, MACAVITY had been introduced briefly into one course, 1.04, "Behavior of Physical Systems," in each of two successive years. In the fall of 1986, the course was taught by Professor Robert Whitman. MACAVITY was presented on an optional basis for use with two problem sets, mainly as an exercise in using the program in order to get some feedback from the students.

Following this exercise, Professor Slater met with the students who had used MACAVITY. He concluded that, while "only a couple of students used it," they enjoyed it and learned from it. On the basis of this experience, however, he also concluded that the first use of such programs ought to be required, for the "overhead involved in coming onto a big computer system" will lead students not to use the programs, even though students who are required to use them say they would gladly do so again.

The impressions we gained from our interviews with students suggest something very different. Two of the eleven students in our sample remembered having used MACAVITY and had opinions about it; another student had heard of the program but said he "had not bothered with it." The two students who remembered using MACAVITY viewed it in a way that was, on the whole, negative. They made reference, first of all, to annoying bugs in the program. For example,

> The second time, I think, we just kept running into bugs, just kept crashing .... It would dump out of the user mode into the inner program, the LISP that the program was written in ...

But their strongest reactions had to do with MACAVITY as a too-simple program:

> ... it can show you some of the concepts [but] ... I mean, I could have taught a six-year old to do what I did on the computer. You know, have numbers, type them in, hit return.
>
> It gets down to the very basics, so once you've learned the very basics, it's easier for you to do it on your own.

As a result, they concluded that the program was really designed not for them but for other, "slower" students:

> It seems like MACAVITY would be almost more suited for people who are a little less mechanical, a little bit slower at learning things.

They saw their experience with the program as "just going through another exercise," just "more exposure to computers," and "not really learning anything new."

By the time these students arrived at the second problem set, they seemed to have decided that the program was not intended for their level of competence in structural analysis. They had developed a theory about what was going on—"just more exposure"—and they began to conduct their own experiments with the program. They learned, in effect, to "beat the system"—that is, to make the program behave not as a tutor that helped them learn how to analyze a beam, but as an assistant that analyzed the beam for them automatically.

> We had to write in equilibrium equations ... and then it would solve it for us, after we wrote in the equations. And then we learned that you could just write—there are certain words you could type in ... [and] it will solve it all by itself. So we gave up on writing the equations.

Their experience with the program had elicited a creative response, but it was one that ran counter to the developers' intentions.

Both the teaching assistant in 1.04 and the research assistant currently involved in MACAVITY's development were aware of these problems. The teaching assistant attributed them, first of all, to the fact that the version of the program tested in 1.04 was incomplete. Second, he pointed out, it contained only the first part of MACAVITY, which he viewed as rudimentary for MIT students in 1.04.

He would not be surprised if these students were to say, "I don't need a computer to teach me that." He thought that the program in its new version—more "handy" and "more powerful"—would be more effective, and that the second part of MACAVITY would not be "that obvious for an average student"—though for a good one, it would still be "boring to use MACAVITY."

The research assistant was explicit, in his interview with us, about the problem of tuning the program to an appropriate level of difficulty. We have to assume, he said,

> ... that students already know a lot. Where do you draw the line? You can't teach addition and subtraction. We have to assume a level. We aim for the average normal undergraduate. The question becomes, is the student dumb, or the program? You can't make it too easy, you can't make it too difficult. We have to decide to do those things that the students have great difficulties with, and that the computer can do easily.

Our interviews suggest that this issue remains unresolved, a view shared by the research assistant. He felt that there was far too little attention given to the educational questions at stake in the program's development, and far too much emphasis on the merely technical challenges—an imbalance that contributed, he felt, to the developers' failure to draw useful lessons from the students' feedback.

In contrast, the teaching assistant was very firm in his view that MACAVITY is a *teaching* program—apparently, because it is *intended* to function as a tutor—whereas GROWLTIGER is "an applications program." As he said about the latter, "You use it. You don't learn anything from it."

Faculty Experience with GROWLTIGER

GROWLTIGER, like a projective screen, is a material that evokes extremely different views. Professors, teaching assistants and students differ, within and across groups, in their perceptions. Their perceptions themselves become actors, collectively shaping the program's meanings in educational practice.

The faculty members who have so far introduced GROWLTIGER into their courses tend, on the whole, to distance themselves from the program.[5] Most of them treat it as an add-on, do not take the time to learn how it works, leave it

[5] GROWLTIGER has so far been used in sophomore, junior, and senior courses in structural engineering. It has been used in 1.105J, elementary lab (Professor Gibson); 1.04, introduction to structural engineering (Professor Whitman); 1.50, structural analysis, steel (Professor Kausel); and 1.52, structural analysis, concrete (Professor Buyukozturk).

out of their classrooms, and delegate its use to their teaching assistants. They do not behave as though they see GROWLTIGER as an experiment in which *they* are involved; hence, they do not see themselves as contributors to the program's development.

For example, Professor Whitman speaks of introducing GROWLTIGER into 1.04, first of all, to "get some usage of computers into the subject" because undergraduates are not sufficiently exposed to computers. He says of himself, "I am not a computer person;" considers his knowledge of computers obsolete; and declares, "I do not even know how to login."

From his perspective, problems inherent in GROWLTIGER's development become a nuisance. He mentions "a perfectly reasonable example" that "caused the system to break down," and admits "I don't know what the problem was." He does not use the program in his lectures. If he were "a real computer jock," he might do so; but he adds, "I'm not; and I'm not in a position to devote the time to learn how I might."

Two other faculty members who have made some use of GROWLTIGER in their courses take a generally positive view of it and indicate their possible interest in developing it further, although they have not done so.

Professor Lorna Gibson has introduced computers into her elementary lab course, 1.05J, in two ways. She has concerned herself mainly with computerization of data collection and data processing. And she has made GROWLTIGER an option for students' group projects in structural design—an innovation with which she is, for the most part, satisfied. She thinks of setting up demonstration experiments, especially in cases of measurement that fall between clearly defined methods, in which theoretical predictions derived from the program will be compared with lab data about the effects of loading conditions.

Professor Oral Buyukozturk has introduced GROWLTIGER into 1.52, structural analysis of concrete, even though the program in its present form is mainly designed for use with steel structures. He sees it as a replacement for STRUDL, the finite element program, which he considers "not well maintained" and more appropriate for use in professional practice where "you don't worry too much about format of the output, the interactive aspect." Buyukozturk plans to have GROWLTIGER modified to include the design of concrete structures, reasoning that, as the technology changes, "we have to keep up with it a little bit so that we can intellectually talk [about it] and make recommendations to graduate students." He treats GROWLTIGER as a tool for analysis and design rather than for "teaching the fundamentals" which, he believes, "has not really been af-

fected." And he doesn't monitor the students' use of the "tool," leaving that to a research assistant who is "in charge of the facilities."

In contrast to Whitman, Gibson, and Buyukozturk, Professor Eduardo Kausel, who teaches 1.50, "Structural Analysis, Steel," has not only introduced GROWLTIGER into his course but has learned to use it, adapted it to classroom teaching, and thought critically about the broader uses of computers in engineering education. These points come out not only in his own comments, but in those of his teaching assistant who underscores Kausel's interest:

> Kausel is more interested in GROWLTIGER. He's used it, he's played with it .... He's used it toward the end of the class. We did this method of approximate analysis, and then he actually ran the instruction on GROWLTIGER to get the exact analysis ... [comparing] exact and approximate analysis.

Kausel tempers his interest in the educational use of computers with a certain skepticism. He worries that a new generation of architectural engineers will become uncritically confident in "these canned programs" and that ever more sophisticated expert systems will be mindlessly used for "the many, many things where ... a machine will never really substitute for a human being." As a practicing consultant, he has seen managers assign complex computer programs to young engineers who "use [them] not only without judgment but sometimes also without even analyzing the program's possibilities," producing "results [that] really are garbage."

These concerns have not led Kausel to shy away from using computers in his courses. Rather, as a way of protecting students from naive use, he tried at first to help them learn structural engineering by *writing* computer programs. He found, however, that they spent far more time on programming and debugging than on engineering substance.

In the end, Kausel used GROWLTIGER in three ways in 1.50. He uses its output in a classroom lecture. He gives students the option of using it to check hand calculations in their problem sets. And he requires the use of GROWLTIGER in a term design project due at the end of the course. This last idea was suggested by his teaching assistant and in the main, Kausel left its implementation to her.

In this project, as she described it, students used GROWLTIGER to analyze and design a building frame. Starting with a "default section" supplied by the program, students analyzed the structure and proceeded iteratively to select members, reanalyze, and make changes, until "member selections match the analysis, everything is safe, and nothing needs to be changed."

Kausel was impressed with the educational effects of GROWLTIGER's use. He noticed, for example, that when the program's limits were unexpectedly challenged, students "got very different answers using frame theory and GROWLTIGER."

> The students were a little bit careless as to how they entered the cross section properties .... They only concentrated on the moment of inertia and they didn't care much about the cross section of the area. So they took relatively arbitrary numbers and, as it turned out, those numbers were too small .... Of course, when you are thinking of a frame, you are not thinking of the cross section ... and of course, it was an unrealistic cross section area .... GROWLTIGER would not use those numbers, and they don't act the same.

This, Kausel thought, was "a valuable educational experience" because "it really teaches how the students have to be more careful about what they do." It is worth noting that Kausel became aware of such educational impacts because he monitored the students' use of GROWLTIGER in their design projects.

In spite of the fact that Kausel stands as an exception to the general pattern of how Civil Engineering faculty use GROWLTIGER, he also shares an important perception with the others. All of them are troubled by the difficulties in making the programs work and by the related demands on them to learn, as Kausel put it, "a new set of rules of the game."

Whitman sees himself as a user bothered by the "constant disruption" produced by continual changes in hardware and software, changes that include the department's removal of his few PCs when it decided to centralize its computer facilities. Whitman notes that his computer knowledge "has recently become obsolete for the sixth time."

Professor Gibson, generally satisfied with her use of GROWLTIGER, nevertheless points out "the only real difficulty in the experiment"—namely that "John Slater is always updating and improving" the program, so that when students try to use it, it frequently does not work. This she finds "frustrating" but, as she says,

> I think that really is just a part of the business of getting it running and started, and once it's all settled and working properly, it'll be good.

Nevertheless, she adds that in her various uses of the computer in her laboratory course, "we have been scrambling to get the thing working at all." Professor Buyukozturk also recognizes that problems are inevitable when a program is still

under development. Such "frustrations," he feels, are essential to computer work. He leaves them, in general, to be sorted out by his teaching assistant.

And Kausel, finally, remarks how difficult it is to deal with the computer when "when the rules of the game change every year or so." You have to "go back to square one and re-learn certain things," which is all right "if you don't have to do it so often." But "if it happens on a regular basis,"

> ... then you finally say, "The hell with it, I'm not going to do it." Because people have other things to do without learning about using computers.

And he adds, so there will be no mistake, "I mean I'm talking about my own experience."

Student Experience with GROWLTIGER

Students showed their awareness of their instructors' attitudes toward GROWLTIGER. Several students said they thought the program had been used in a way that was "forced," "contrived," a "something extra" that was "thrown in" without a corresponding change in course design. They mentioned professors "who do not use the program themselves, or really know about" or use it in their lectures—although here, some students made an exception of Kausel's use of GROWLTIGER in the classroom.

One student said that "in most classes ... it was just like they had thrown in the computers,"

> They didn't really need them, but they just sort of threw them in so that you'd get some idea of what they were, and get some exposure to them. But it wasn't the focus of the course, or anything. It was just something extra they didn't really need.

Most students considered that their best experiences with GROWLTIGER were in Kausel's course, 1.50, and to a lesser extent, Buyukozturk's 1.52. They particularly appreciated using the program in the term design project. And they recognized the teaching assistant's special contribution in this regard.

> If she hadn't pushed, I don't know that we would have used it. I know that in 1.50, the professor was all in favor of it, but he didn't know much about it .... I think that mostly the TA knew about it and she's the one who promoted it. Same thing in 1.52.

On the other hand, the students who had taken 1.50 did not think the program had been very helpful when used to check problem sets. The teaching assistant had encouraged the students to use the program in this way, for reasons

that echoed Kausel's concerns about the mindless use of computers in industry: "Right now they would be using the computer to check hand calculations, but in industry you really need to use hand calculations to check the computer results." Students, however, found it "too much trouble" to use GROWLTIGER to check their problem sets, especially when, as one of them said, "it does not matter that much if I had the right answer." And the teaching assistant, aware of the student's close calculations about the use of their time, commented that "there was a lot of work to do [so] the last thing they're going to do is walk from their little dorms all the way to Building 4 just to check their answers."

As was the case for MACAVITY, students' experiences with GROWLTIGER diverge from the developers' intentions in some very important ways. But in this case, student experience is, if anything, richer and more positive than the developers anticipated. All students agreed they found it easy to learn and easy to use. Only the two transfer students reported temporary difficulties, because GROWLTIGER was their first contact with Project Athena, and it was here that they had to learn UNIX commands and the habits of Athena's workstations. However, their discomfort had as much to do with what one student called the "culture shock" of MIT as with GROWLTIGER *per se*. In addition, all said they enjoyed using the program and that it was helpful. Their descriptions echoed Slater's characterization of GROWLTIGER as a useful *tool* that does the work of analyzing structures for you and allows you to try out many different possibilities, sparing you the need to carry out long and repetitious calculations.

But how students characterize the "tool" reflects important differences in their personal responses to the program. Some celebrated GROWLTIGER's ease and efficiency:

> ... it can simplify my life because [it] just spits the answers right out at you ...
>
> ... it analyses it for you. You can't ask for much more!

Others were concerned that GROWLTIGER, precisely because it spared you the burden of analysis, did not *teach* you the "basic concepts" that you need for structural analysis. On the contrary, it could actually keep you from discovering "what's going on" in analysis. One student pointed out that, although the program serves a useful purpose for problems that are larger and more complex, for smaller problems "you can probably learn more doing the analysis by hand."

> If you're sitting down and actually doing a problem set, you're spending the same time, except you'll probably learn more because you're not running into bugs .... You're not typing ...,

> you're not trying to weave your way through the way someone else thinks when they wrote the program. In the time allotted, you probably learn more on your own than you do with the computer. Because, on a test, you don't have it.

In this student's comment, as in many similar ones, the advantage of "doing it by hand" in order to *learn* more is closely coupled to a cost/benefit calculation in which "time" and "grades" figure prominently.

In their appreciations of GROWLTIGER, many students mentioned other advantages that had been central to Slater's intentions. In a number of instances, students said that GROWLTIGER lets you work on problems that are real and practical—complex structures that you couldn't analyze any other way. As one student put it,

> I don't think you could ... tackle any real-life problem without the computer .... You'd have to stick to the types of things that you do in class and on tests: really simple problems. Because anything over, like maybe two stories, you can't do by hand .... I mean, you can, but it's so cumbersome that you wouldn't bother. So you get to actually do a real-type problem.

Another student was amazed to discover, in his interview, that in the 1950s people actually designed large, high-rise structures without the computer. To students like these, it meant a great deal to be working on *real* problems, as opposed to the "simple" and "artificial" problems they were used to seeing in the classroom.

Many students see GROWLTIGER primarily as a tool for *design*, and for more than a few of them, their experience with GROWLTIGER seemed to compensate for a perceived lack of attention to design in the department. One student noted that

> ... in the classroom [in 1.04], we learned how you analyze the thing, but we'd go to the computer to learn how to design, and I think it kept [me] interested in what was going on in the class. It was a lot more fun to design!

The teaching assistant in 1.50 elaborated on this point:

> ... I can sense that there will be students who will wish that they knew more about design, and probably most of them, because ... a good basic undergraduate curriculum in civil engineering should introduce them to design, especially at a place like MIT who's going to turn out the high caliber people. I mean, they'll be the leaders in engineering in thirty years ...

> and they should have some kind of rigorous academic training about design methods.

For this individual, GROWLTIGER's special merit comes from the fact that Slater, its main author, is primarily a structural engineer, a designer, and only secondarily a programmer.

Many students went beyond the description of GROWLTIGER as an efficient analytic tool useful for the design of large and complex structures. They spoke, as had Slater, about the program's ability to help you increase your understanding of and intuition about structural behavior. They identified a variety of ways in which GROWLTIGER can fulfill these functions, emphasizing themes of iteration, visualization, theory-testing, and surprise. Some students were struck with the fact that, because GROWLTIGER makes it possible to produce many fast iterations in the design and analysis of variations in a structure, without the tedium of hand-calculation, it can give you a "feeling" or "intuitive sense" of structural behavior. This was seen as being due to several factors. First, using the program gave one "time to think," free from the need to worry about calculations. And the program also made it possible to observe the effects of multiple trials. Both themes are present in this student's observation:

> When you're doing it by hand, you're so worried about numbers and things, and just seeing if you get reasonable numbers, and working through calculations. You don't really have time to think as much about what you are doing. With GROWLTIGER, I was sitting there and I had to minimize the deflections at one part of the building, and I just kept trying different things, and seeing what effects they had, and you got a much better sense of the interactions between the members.

Other students emphasized that GROWLTIGER enhances intuitions of structural behavior because of what it allows you to *see*. Many students made this point, referring in the process to different kinds of visualization. One student spoke of seeing "deflection patterns," which he called "a learning experience in itself:"

> ... in class, you don't really always see ... deflections in relation to the moments. Although there were a few times that it was cited. But when you look at different beams on GROWLTIGER or different frames, you get an idea of the relationship between the shear of the moment and the deflections, and the curvature relationship.

Another student said that when he actually sees where there are small and large moments in a structure, he knows where to put the reinforcements and "actually

gets an idea of what's going on," whereas "If I show you these numbers, I suppose they won't be very useful to you." And a student who was not sure how much you could learn with GROWLTIGER that you could not learn as well "without the computer," was nevertheless impressed with "seeing pictures of deformations" that you wouldn't learn otherwise.

Some students thought GROWLTIGER's main importance lay in its relation to the theory of structural behavior. Some of these students took a position similar to the teaching assistant's, who thought that GROWLTIGER was just "an applications program" from which you "learn nothing." They argue that you need "the basic concepts of civil engineering" in order to *use* GROWLTIGER, although its use can reinforce these concepts.

On the other hand, some students note rather sheepishly that even though you are supposed to "know the theory" from classroom lectures, it is quite a different kind of "knowing" that you get from actually seeing how structures respond to variations in design. And one student in particular—William, to whom we shall return—was very explicit in his description of an illuminating experience in which he had used GROWLTIGER to help him generate and test theories of the unique, complex structure he had chosen to design.

Many students were able to remember instances where they gained new insights into structural behavior because GROWLTIGER had *surprised* them—had produced puzzling counterintuitive results that made them try to figure out what was going on. In this, GROWLTIGER enhanced the educational experience in a way that had not been anticipated by its developer.

There were many references to such surprises. One student spoke, for example, about discovering the counterintuitive behavior of his structure as he explored the stiffness of columns and floors in his 1.50 design project.

> The design formulas are saying "this is an efficient section to use," and I put it in the computer, and it deflects too much. So I have to see that, increase my section and stiffen it more. And I asked the TA and said, "What the hell is going on? How come it doesn't correlate?" And she says, "Well, because this is such a complex system, there's just so many degrees of freedom ... that it just needs to be stiffer." And that's one thing that I wouldn't have realized had I not used GROWLTIGER. So that was good.

Another student described his reaction, in 1.04, to a discrepancy between his hand calculation and GROWLTIGER's:

> I wanted to do some checking, so I did it on GROWLTIGER and it gave me some answers. But I'm still not sure if I formulated the problem correctly ... if I gave it to GROWLTIGER correctly. The answers are very close, but I'm not sure if they are 100 percent.

The seven students who reported such puzzles seemed to see them as their most educationally interesting encounters with GROWLTIGER.

Some of these students became aware of the significance of surprises in the course of their interviews. At first, they gave the usual instrumental description of GROWLTIGER and only later, after probing, began to recount their more unusual experiences.

Samuel, for example, first described GROWLTIGER as a tool that allowed him to make and evaluate quick changes in a structure. Upon further questioning, he began to talk about "visualization" and "feel."

> I applied a windload and I saw it lean a little and I noticed that because the building was so vastly cantilevered, only the middle third of it was supported. I saw the continuity of how it would have to behave if it weren't going to fracture and fail catastrophically. So in that way, just by showing a visual representation, and allowing me to change different beam sizes and loadings, I could get more of a feel of how an actual building behaves and what actually happens.

When the researcher asked this student if he had encountered any surprises, Samuel mentioned how "the stiffness affects adjoining members,"

> so that if I wanted to get rid of some deflection in a floor girder, I could stiffen up the wall columns .... I understand that bending moments go around a solis joint .... [Then I saw that] by making some columns a little bit wider, *without necessarily changing the weight, or even lowering the weight, I could make the building stiffer.*

So this student passed from describing GROWLTIGER as a tool for the quick evaluation of structures to the idea of "visual representation" that gave him a "feel of how an actual building behaves," to surprises that yielded new insight into structures. All the while, he kept saying, almost apologetically, that "theoretically he knew" and "it should seem obvious." Finally, he concluded that the lessons he had learned from "surprises" had now become truisms; and he observed of the theories he had learned in class,

> I had learned them all right, and I knew them to be facts, but I didn't know them to be actual and true until I saw them working in this structure!

Although a few students touched on a wide range of themes—as with Samuel's successive self-descriptions—the more usual pattern was that different students emphasized different issues. GROWLTIGER's meaning seemed to vary with an individual's attitudes toward computers, engineering, and life at MIT. Here, it is helpful to think of the computer as a projective screen for other concerns, for a range of things about a person.

Part III. Diversity of Student Experience with Computers, Engineering and MIT

Programmers, Users, and Avoiders

The eleven students we have interviewed seem to fall into three groups. There are *Programmers*, three students who enjoy writing programs, see themselves as good at it, and do a lot of it, although they say they probably would not want to do it for a living. There are *Users*, five students who say they like to use computers and are comfortable with them, although they do little or no programming. And finally, there are *Avoiders*, three students who say they have negative feelings toward computers, "hate them" or actively seek to avoid them. Nevertheless, they acknowledge that avoidance is not a feasible strategy at MIT, and declare that computers may be "useful tools" for some purposes.

The three groups do not differ in their pre-MIT experience with computers. In high school, all of them had learned BASIC; and some of those who had learned and used more than one programming language in high school nevertheless became Users or Avoiders at MIT. More decisive for the MIT students' attitudes toward computers was their experience with an introductory computer subject, 1.00 or 6.001—an experience reinforced, in some cases, by a later exposure to programming in a job or field placement.

* Avoiders

Ted, who is typical of the Avoiders, expresses extreme ambivalence toward computers:

> I came in [to 1.00] knowing pretty much nothing. I remember the first program I handed in ..., he gave me a 30 because I didn't put remarks in and commas and, you know, I didn't know any better ....

On the one hand, Ted says that he "didn't feel too bad" about them; on the other, he "didn't want to program them at all." Once the program worked, he mused, "you could do a lot of fun things with it; [but *getting* it to work was] a pain in the neck." So, he decided, "I'd just use the programs." And he was greatly relieved to discover that this could be a viable strategy at MIT.

> I remember asking a guy who was a junior at the time, because I think I was saying ... I hate computers ... 'cause that's all I knew about computers was programming at the time and he was like "You better get used to it, you're going to have to use them for the next couple of years." And then I asked him, "Am I going to have to program all my own stuff?" And he said, "Most of the time they'll have a program for you. You just have to use it to get data." So I'm relieved.

Raj, a student from a developing country, took 6.001 and dropped it.

> The main thing in LISP was recursion—how to use the function to act on the same function—it worked on itself. It makes the program very short, but it took me some time to understand it, and I'm still not so sure. I hear from my friends that once they understand that concept, it's so simple ... [but] then I dropped the class.

He later took 1.00 "to the end" and found it "more mathematical," "more engineering," and "more straightforward." Yet in later courses like 1.02 and 1.03, he decided not to do the problem sets on Athena, but used BASIC instead—the "freshest and easiest to pick up again."

Programming he sees, above all, as "a time sink," especially for him because he is "one of those people who take longer than average." He sees people "spending a lot of time" but "not as much time as I think I am spending on it." At one point, he had made up his mind either to "get away from it" or "to learn how these people are doing it so fast." But he seems now to have decided that it is not worth his while to pursue the matter:

> If I don't see the point in it, I'll say, well, I don't want to put so much time and energy into learning.

Raj describes himself as "honestly trying to avoid computers." Nevertheless, he recognizes that "MIT is not the place to run away from programming." He

has learned that, "if you want to be in areas that you like, you've got to use computers." What he wants, then, is a job where "programming is an aside ..., a tool rather than the thing that I'm doing."

For the Avoiders, debugging programs is an ordeal that induces feelings of "panic" and "hopelessness" strong enough to cause them to avoid programming. Ted's comments were typical. In 1.00, his first encounter with programming at MIT, he "spent a lot of time in there," but

> I just couldn't get the bugs out .... Towards the end, I wouldn't try as hard ...; it gets a little more hopeless ...; I ended up copying.

When he first began to work on a problem, he was not too worried about fixing it, but then

> ... after you've been in there for a while, it gets on your nerves, and besides you don't have too much time to get it done. You're running out of time, so you get a little frantic, I guess.

For Avoiders, GROWLTIGER is a welcome relief from programming. It subordinates the computer to the practical, large-scale, socially relevant tasks of Civil Engineering. For Avoiders, its principal merit is that *it gives you the answers.* But it also lets you *design*, sparing you the tedium of hand calculations, and it allows you to *see* what is actually going on when loads are applied to structures.

*** Users**

Students in the group of Users talk more positively and less apprehensively about computers. They describe themselves as "comfortable with computers" and find them "self-explanatory." One student in this group said he never had any trouble learning to use computers and could "patch together little programs." Indeed, he had taken a summer job with a personnel consultant and had written a small expert system on this job—what he called "a computer embodiment of the man I worked for." Yet he says "I don't really program myself," and has no interest in a career centered on computers. Another student liked 1.00, even though he found it a "time sink." Another who found 1.00 "enjoyable" offered this definition of computer literacy:

> ... you should be able to write a very basic type of program, outside of this particular department, just in general terms. It means being able to walk up to a computer, use software that's already in it, basically. Even if you've never had exposure to the software, go on, be able to use it.

Acquiring this sort of literacy meant "just being exposed to a lot of different programs." On the other hand, this student "hated 6.001" and could not see "spending hours in front of a computer screen."

Still another student had taken a year-long UROP project, working with computers in a materials testing laboratory, and had become "disillusioned" with them. His job had been repetitive—he had learned to transfer files, "a lot of grundge work," and he could not see himself at the computer for "more than five hours at a time."

These students—several of whom, in spite of their disclaimers, seem to have acquired considerable proficiency in programming—are characterized by their position that computers are interesting, as one student said, "not as their own end but for what you can do with them." They should not "dominate courses" but should be "directly related to the course, not just something to do with computers." They are "a tool you're going to have to use wherever you go"—a useful tool, but merely a tool. Students in the Users group refer to their frustration with bugs as their main reason for dropping 6.001 and for having as little as possible to do with programming. One student, for whom bugs present "real problems," remembers how he became "upset" and "mad" when, on several occasions, he encountered program bugs he was unable to fix. He became frustrated when he was stumped by a difficulty that "everybody seems to be able to overcome" in "what should have been a pretty straightforward application."

> I've always done well in school, and so if I encounter a situation where I feel I'm not doing well, it's like personally frustrating. It's bothersome that I can't overcome this difficulty that everybody seems to be able to overcome.

It diminished his frustration, but only a little, to walk away from the computer and find out the next day that everybody else had had the same problem.

Another User referred to the frustrations with "syntax errors" that induced him to drop 6.001. For him, these were an *impediment* to learning:

> ... with any computer language, forgetting a semicolon will set you back half an hour if you don't realize it quick enough. And that half hour, to figure out that there is something wrong just in the way you wrote something, is wasteful to me. I could be learning something; I couldn't be debugging.

Users, unlike Avoiders, do not see Civil Engineering and GROWLTIGER as an *escape* from programming. Rather, they focus on GROWLTIGER's ability to let you see what's going on, test your grasp of structural theory, develop your

intuitive understanding of the behavior of whole structures, and engage without drudgery in process of design.

* Programmers

The three students we have classified as Programmers see themselves as good at it and do a lot of it. Gabriel describes himself as "very comfortable with computers," said that he "could see doing a light amount of programming—just programming myself, to help the problem," but "could not see doing programming ... for a profession or a career." Although he doesn't do a lot of programming, Ali writes programs for particular uses—finite element programs, programs to reformat files—but also enjoys just "playing around:"

> Yeah, I wrote a few programs just to play around with the computer. With my small computer, I wrote a program of two people playing tennis .... [I] play around with characters, draw people, stuff like that.

He "likes computers a lot." He spends "hours and hours just trying to figure out how it goes ..., then when you learn to just ... It's great!" Paradoxically, however, Ali adds, almost as an afterthought, "Personally, I don't program. I don't do a lot of programming."

William's first encounter with computers at MIT, course 1.00, enabled him to "gain some confidence with computers." With Athena's "comprehensive system," he found "... not only that I can perform certain operations, but I know that if I have a problem there would be a way to solve it and to get around it." Of all the students interviewed, William is the most prolific and proficient programmer:

> I write a lot of programs I use. I have about 60 programs of my own. Very little programs. Just to work around, just to save matrices or to solve ... specific equations, because those are things that are very tedious.

In contrast to Raj, William's "costs and benefits" analysis leads him to spend a lot of time programming:

> If a problem set is going to take me three hours without the program, the only way you write a program would be if the program took me an hour. So one third of the time. Because you've got the program done and then you use it for half an hour, and then your problem set is basically done.

He has also written larger programs—in one instance, a 500 line program that he particularly "cherishes." His experience with a summer job reinforced the positive attitudes he had first acquired in 1.00. He liked working as a programmer because, as he says, "I like to think:"

> I like to think, and therefore I like programming. For some people, programming is very dull work. For me, it is not, because you have to think, you have to be efficient, first of all ... I don't know, it's just a different way of thinking. Who knows what I like about programming? I even like getting rid of bugs. I mean, that is the worst part of programming, but I still like that ... I probably like it because I can make a machine who thinks faster than a person could do, and I am the creator of such a routine, and that's probably why I like it.

For the Programmers, bugs are, at worst, minor irritations to be overcome through persistence; at best, a source of pleasurable puzzle-solving and a way to build one's confidence with computers. In a dialogue between Gabriel and Sydney, when Sydney states that bugs discourage him, Gabriel notes that, on the contrary, they spark his interest. Gabriel recognizes in himself "a talent for not making syntax errors;" in fact, such errors make him "a little more interested" and solving a programming problem makes him feel

> ... very proud when I can do something, whatever. Just to be able to take it and do it and say, "Yes, I've done everything that was required and everything that you need to do for this problem."

Ali, remembering the "hours and hours" he spent writing a finite element program, remarks that he persisted until he had fixed the bugs, "just because I want to have it solved." And in the midst of William's general celebration of the computer, he observes not only that he actually likes fixing bugs, but that debugging is the way he has gained his confidence:

> ... knowing the mistake that I made before, and then I have to get out of it, gave me this confidence [that] if I made the same mistake, I wouldn't have any problems getting out of it ... so it's important that you get lost, that you make mistakes in anything, because that will give you strengths to get out of it.

The three Programmers spontaneously reflect on their styles of programming. Gabriel, for example, describes himself as "working backward to a certain degree:"

> ... I have to come up with this value. And to do that, what do I need to get that value? Well, I need these things. Well, how do I get those things? I need to make these functions. So [I do that] just in my mind .... Then, when I start doing it on paper ... actually no, I guess I pretty much just start writing. Like you have all your variables. Like I'll write and then, as a variable comes up, I'll stick it in there .... I just try to organize it in my mind and then write it out.

Ali thinks of "transforming a problem into a computer code" and speaks empathically of "what the computer can understand."

> For example, I have a problem, okay? I need to solve it. It's just not ready for you to put on the computer. You have to transform it to a computer code which sometimes is not very easy to do, because you have to think ... this is how we understand it, but the computers can understand it this way or this language has to be in this format. So you write it so there's some certain flow of logic in it, so that you have to think step-by-step how the computer is going to interpret this thing, and you have to accordingly format your problem of programming.

William, finally, produces an elaborate description of his way of programming—one that he seems to have discovered in the course of his interview with the researcher.

> And if I cannot do that [use existing programs], I ask myself, "Can I write a program for this? ...." And I think. I think while I swim ... and when I decide to do a program, I just do it. I just go there. Now basically, what I do first is ... try to write it as quickly as possible. Without thinking. When I do it, I just write the steps it would take my brain to do it. That's very crude. And I don't usually keep those programs for the most part, because those are very inefficient programs. And then I go, I go swimming or biking, ice skating, or whatever. And while I do that, I think about the program, how can I make it more efficient? ... And I say, "How can I make the computer do this in such a way that it will take me 10 minutes to write a program, that the computer will not need that much information to do its data?" Then I think, "How can I make sure that the computer doesn't overwork, because of computer time?" So I try to make it more efficient computer-wise. Then finally, "How can I drop possible bugs?" And I think also where the point is where I can make mistakes? And I try to be careful in those places, or just not do them at all; just go around it .... And then I come back and I rewrite a program, and it's usually done. That's how I do it.

When asked where he learned to program like this, William refers first to 1.00 where he learned the "ethics of writing a program—I mean, top-down programming." But then he begins thinking of his earlier education in architecture:

> ... this ... is what I call catching the program. Just the first catch. I don't think anybody has told me that. But that is basically what I've learned by doing it. Most of the things I do are like that. I first do a sketch. Just as I mentioned to you, I like architecture, and I was pursuing a degree in architecture, and the first thing that you learn there is just draw a sketch. So that's what I do in everything .... The first thing I did was I learned it just very crudely and then I perfected it. So basically, maybe I had that before. Maybe the way I understood that was the way for me to write a program was by doing it. So this is the first time that I realized how to write a program. It just comes to me naturally.

Gabriel, Ali and William take pleasure in their mastery of programming as a craft. For them, it has become a medium for spontaneous work and play, a source of pride, and an object of reflection. They tend to treat GROWLTIGER as they treat computers in general—they are interested in discovering *how things work*. They are intrigued, rather than disturbed, by surprises, and use them as springboards to insight. Some students who like to program see bugs as enemies to be relentlessly eliminated. These programmers have a more flexible, "softer" approach. They like to *play* around with computers and they like to use GROWLTIGER to *play* around with structures.

This is William's description of his way of using GROWLTIGER:

> I just go to GROWLTIGER as I go to Athena, I mean, to any other Athena programs .... I sometimes will play with it, because Civil Engineering is my field, and I think it teaches you a lot. So the program is very valuable.
>
> ... in engineering, there are things that are intuitively obvious, and there are things that are not. And sometimes things that seem to behave in one way, behave in a different way. And the only way to find out which way something complicated behaves is by working a lot of numbers and spending a lot of time, and then finding the answer. Or going to GROWLTIGER and just set up the problem and ask GROWLTIGER to solve it. Not necessarily because you want to come out with the answer with a force, or you want to come out with a moment ... but just to see how the system as a whole behaves.

William described how, in testing GROWLTIGER, he had also tested himself. He had chosen a very complicated structure with a curvature that

GROWLTIGER could not analyze. When he had approximated the curve using small linear members and obtained the deflections, these had not seemed obvious to him. As he tried further variations, he ran into a series of surprises. When he eliminated a column, for example, he found that the forces in the entire structure decreased. Not believing what he saw, he replaced the column, and then decreased its stiffness in small increments, until finally he believed the evidence provided by his own eyes. And he explained this result as follows:

> ... what happened is the configuration of the frame led me to believe that if a frame could be somehow counterbalanced, this column was not needed ...; the frame was almost balanced. So the three tier columns would be able to hold the frame just fine. But the addition of the column on the left-most side was just keeping the frame from being totally balanced. It was taking more stresses and just creating bending stresses on the center columns. And the tension of the stresses is summed. So that was my first theory .... But the only way to prove it was to play with GROWLTIGER and see if that was in fact true. And so GROWLTIGER convinced me.

He also found a discrepancy between the results of his approximate hand-analysis and the exact analysis he got with GROWLTIGER. Again skeptical of GROWLTIGER, he went through the approximate analysis until he discovered that it left out certain deflections that GROWLTIGER did take into account. As he said,

> ... there's a redistribution of forces when you have numbers bending and interacting. So everything is redistributed. Approximate analysis doesn't take care of that.

At that point, as William put it, he had

> three results that matched. The ones I used by hand, the ones that I did by decreasing column stiffness, and the final one without a column in the frame.

Finally, when he began to play with the loading conditions, William found that in a certain situation, the structure would collapse if he did *not* add the column he had previously decided to eliminate because it reduced forces in the structure as a whole. As he said,

> They just give you a building, and you assume that you're going to have certain kinds of loading conditions, which might or might not be true. So you have to take into account what would happen if the loading conditions changed.

He needed the extra column, then,

> ... because of the problem of *stability*. I need that column anyhow, even though it will increase the forces.
>
> When I changed the loading conditions, what I gathered was a frame which was not counterbalanced. When the number was not counterbalanced, that column that was inducing additional stress, which was the one that I was trying to get rid of, it was needed for stability.

Just as William plays with computer operating systems and programs to see how they work, so he treats GROWLTIGER as a virtual world for playful experimentation.

Getting Help

All students described help-seeking as something they had to *learn* to do. Indeed, their diverse strategies of help-seeking reflect their different attitudes toward computers as well as their different levels of proficiency.

Avoiders, for example, were the least inclined to seek help either from faculty or teaching assistants. They tended to keep away from teaching assistants because of their feeling that recitations were "a waste of time" or because they "didn't want to go in there and look bad." As Ted put it, for example,

> [In recitation] there'd be a couple of guys and they'd be real computer wizards and they'd be asking the teacher, "What about this? What about this?" ... It wasn't very practical ... and I don't know any of that stuff at all .... I didn't want to go in there [to the TA] and have no idea what's going on because it makes me look bad ....

One of the teaching assistants confirmed this point. She had observed that poor students who are "scared" and "don't understand" are the ones she sees least. The students who are most frustrated by bugs are also least likely to seek and get the help they need.

Programmers, by contrast, seem to have learned when and where to seek help. Ali notes, for example, that he has learned to "give it some time" before going to the teaching assistant for help.

> ... I don't like to go to the TA unless I really get stuck, because sometimes it might be an oversight—a very simple oversight—and so I leave it for a while and look at it later. I don't find it difficult to go and ask him—he will be ready to help. But from what I've experienced in programming, sometimes you have to just give it some time before you give up on something.

And William, the student who speaks most cheerfully of his encounters with bugs, is also the one who expresses the greatest willingness to seek help and identifies the widest range of people to seek it from. He has learned to treat certain teaching assistants as computer mentors, and he declares that he is perfectly willing to go to the faculty, even when he has not taken a particular professor's course. He feels that they have a responsibility to help him because "they have to encourage the usage of the [Athena] system and they have to be able to back it up." He does not find professors intimidating and "feels comfortable with them lots of times." Nevertheless, despite their differences, there are certain constants in students' comments about their ways of getting help. All of them, for example, name fellow students as a preferred source of help. They turn frequently to upperclassmen, to students who had established a reputation for competence and willingness to give advice, or just to students who happened to be around at the time help was needed. Many students talk about the advantage of working at the computer together with other students in project teams and problem-solving groups.

All students make reference to teaching assistants, whom they nearly always portray as accessible and helpful. Several students mention a particular teaching assistant to whom they always look when they have a computer problem. And on more than one occasion, students refer to a subject as especially difficult "just because I didn't get along with the TA."

On the other hand, almost all the students interviewed reported their disinclination to seek help from their instructors. One student observed, for example, that he would never go to his instructor for help with GROWLTIGER, because

> ... we always got the impression that he kind of thought design was—well, kind of a waste of time, sort of a very menial task. He was always much more interested in analysis.

A second student said that he would not go to his professor for help with problem sets because the professor "never talked about the homework." A third student admitted that he would not ask a particular professor for help because "I just don't know item one on the topic, and [the professor] would have to come *way down*." Other students echoed these feelings, saying that they like to keep their distance from professors in order not to waste the professors' time. In one instance, a student offered a detailed account of the signs by which he had learned to recognize instructors who distanced themselves from student problems. He began by observing that he found teaching assistants to be "far more available" than professors who are "very hard to get hold of." The professors convey an attitude, he explained, that

> ... seems to let you know that they're busy, and they're willing to help you, "But I hope it doesn't take too much time." I don't blame them for that at all.

When the interviewer asked this student how he picked up clues to professors' "attitude," he said,

> It's like a hurriedness in their voice, or they're busy doing something else, and let you talk to them, and they're only half listening, and they know the answer right away because they understand the material, but when you're trying to explain something you kind of expect someone to listen to you so that they understand where you are.

You get an "intuitive feeling" that

> ... this guy is really busy, and he's helping me, but he really wishes it would be something quick. So there's a kind of negative feedback there. It's not expressed, but it's still present ... I've discussed the same thing with other people who feel the same way.

In the worst case,

> ... the professor says, "Only talk to me two hours a week, and if I'm not here, then I'm busy." If they say that to you right up front at the beginning of the course, then you're definitely not going to go to them first thing to get help, because you know that they don't want you .... There are people like that. You can't do anything about it.

Several students made it a point, however, to exempt Professor Slater from the general pattern of faculty distancing themselves from students. They noted Slater's accessibility to them whenever they had problems with bugs in GROWL-TIGER. They appreciated the "bug reports" they could send to him, and the prompt responses he gave them. And one student noted that Slater was available for any problem he had with Athena:

> He has always encouraged people. Even though I haven't taken a course with him, I know he is a big friend of Athena. So, even though I am using it for some other course, even if it is not his, I can always ask him and I'm pretty sure he can do it.

Slater, this student adds, "taught me how to fish instead of just giving me the fish."

> ... he just came by and I saw him and quite frankly, I didn't know who he was. And I just asked him and said, "Do you know how to solve this problem?." And he said, "Sure!" And he showed me how to do it. The nice thing is ... he told me what to do, he didn't do it for me ... and he spent a good half hour with me, just teaching me how to go through the procedure.

Patterns of Individual Adaptation to MIT

We have noted that students' initial experience of dealing with computer bugs contributes to their attitudes toward computers; and we have shown that their attitudes toward computers influences, in turn, their attitudes toward Civil Engineering, their responses to GROWLTIGER, and one might even say, their sense of self-worth.

Now we turn to three individuals who illustrate how experiences with computers are enmeshed not only with views of self, choice of specialization, and attitudes toward engineering, but with responses to the larger MIT environment. The point is clear: experiences with computers are not an isolated element in educational life—the computer presence is a key actor in how today's student "constructs" the MIT environment.

Oliver describes himself as someone like "80 percent of the people who come to MIT thinking about Course 6," but in the end, turned to Civil Engineering.

Pressed for money, he has crammed as many units as possible into his first two years; and he plans to graduate in seven semesters. He has learned to be very careful about the way he spends his time. Programming computers, he discovered in 1.00, was frustrating but, above all, time-consuming. He knows that he can solve problems with the computer if he "buckles down" and puts in the time, but much of his thinking has been governed by the need to *save* time. So, he says, "I won't put an inordinate amount of effort into anything [if] my utility function doesn't make it worth my while." If he has a problem with the Athena system in Building 4,

> ... and can't get immediate help, it's not worth my while to wait for it; I'll just go home and come back another time.

In Oliver's search for part-time jobs, he decided not to "spend time trying to find [a suitable UROP project] because my more immediate need was to be able to pay my rent and my ... tuition bill."

Oliver sees himself as a "type A personality." And he sees himself in need of reform, trying to "leave things alone and just going away and not getting too

uptight about them." Yet he knows that he tends to be "driven by the problem at hand" and he has learned to be wary. He was alarmed, for example, by what he saw of students in Course 6. He "saw people coming out of some of the [EE] courses and they were ... so run down and worn out," and he said to himself, "I don't need all these hurdles!" He decided, therefore, to avoid all the "theoretical courses" with their competition. The choice is rationalized as avoiding things he has "no intuition for," but at base, the choice is self-"therapeutic."

Oliver describes himself as someone who likes to build things. In high school, he had designed and built an A-frame dog house, a brick barbecue in his backyard, a new driveway, and he had watched his family's house go up and "understood how." Civil Engineering—introduced to him by "a guy in his fraternity house who double majored in Civil and Course 15, and is now making $50,000 three years out"—appeared to him as something "real," something that would "give me practical knowledge," and "something I could relate to every day." It was the "basic knowledge" Oliver acquired in his Civil Engineering studies that equipped him, last summer, to manage the renovation of the building in which he lives. Here, he says, he was able to apply

> ... a distillation of all I had been experiencing, all I had been shown, and it was working successfully and allowing me to function in what I felt was a fulfilling role to get this project done.

This is what Oliver had been looking for, and he was "happy for that." Now, as a senior, he looks forward to a five-year contract with the Navy, in their nuclear propulsion program; but at the end of that contract, he hopes to return to civil engineering.

Bert had learned BASIC and FORTRAN in high school where he wrote tutorial programs in math and science. But he stopped programming in his junior year because

> I lost all my disks. It was just something that I got into. I stopped at the end of junior year, and after that I just never got back into it.
>
> I could have been a lot different. You never know. Things like that happen .... I didn't think at that time it was meant for that to happen. I don't necessarily believe in predestination, or anything like that. But it just happened that way.

His experience at MIT in 1.00 convinced him that "I didn't really have an affinity for computers anymore,"

> ... so that the simpler programs I liked and didn't mind doing, but when it came to like integrating bigger programs into bigger networks, I never really got into that as much .... It doesn't catch my interest as much as some people in this place .... But I just have restrained myself from spending too much time on programming. Being that it's just not in my personality to, like, be in front of a screen for too long. I like to use software programs that help out with solving civil engineering-related problems .... So it's useful in that respect. And that's all I can really say about programming.

A summer job in which he was "just like off by myself programming" reinforced these judgments:

> I would just get stuck sometimes and wouldn't get anywhere. And that just like ... kind of told me that I wouldn't be very good as a programmer also .... You know, it's some little detail with the system that causes these problems. And then you go from one system to another, and there's another whole set of details that you have to learn ..., and that kind of killed my spirit .... Dealing with computers. It's for a special type of person, as far as I'm concerned.

Now, Bert feels, his original liking for Electrical Engineering has been "overshadowed" by his liking for Civil. He discovered that it is just "not in my personality to, like, be in front of a screen for too long." He is not "the very intense person who pays more attention to detail," but is more the type of person who

> ... would take a chemistry set and throw the chemicals together. Not necessarily read the book, but just to find out what would happen.

And so, he declares, he has "restrained [himself] from spending too much time on programming."

Bert has found the pace and competition of Course 6 too intense, and the contents of its courses too "mathematical" and "miniscule." In rapid succession, he had considered Physics and Aerospace Engineering. But Physics he rejected "because I was a little more into engineering." And Aero seemed "a little far removed ... just, from, I guess, society, I don't know."

> I mean, planes are important, but it just seemed a little far removed.

Civil, he felt, "goes along with my personality." He finds it "less high-strung, fast-paced." What appeals to him in Civil are "real jobs," large-scale projects like

dams, but above all, involvement in design. For engineering, he feels, is "about designing things,"

> ... just designing things to the greatest potential, minimizing costs and material usage, and trying to find better materials to meet those ends. And that's basically what it is.

GROWLTIGER's great appeal for Bert is that it allows him "create a structure" when he "feels the urge," and spares him the tedium of analyzing it by hand. GROWLTIGER "makes it much easier" to learn about such basic engineering elements as trusses, chords, panels, materials properties, and moments of inertia. It is a "first introduction to design" that allows you to see how things work—to

> ... look at animations of deflection of structures. You watch the progressive deflection and watch it spring back ... when you look at different beams on GROWLTIGER, or different frames, you get an idea of the relationship between the shear of the moment and the deflections, and the curvature relationship.

Bert aspires to a job where "I would be the chief engineer and the person that's doing the major work." This seems to him "more of a job" than "working on miniscule things." A *big* thing like a structure will always be there.

> So it's just something that, you know, for the rest of my life, I'll be able to say, "Well, I did this with other people, or by myself." ... and to me, that's more substantial.

A "top-down," "divide and conquer" style of programming characterizes some programmers; Sherry Turkle has called them planners. The more interactive, playful styles of William and Bert are more characteristic of what she calls bricoleurs or tinkerers. For them, interacting with a computer is more like a conversation than a monologue. The style of appropriation of the computer technology is highly personal—as individualized as personality and professional aspiration, and we should add, enmeshed with both.

In Michel's story, we get a more poignant account of the ways in which loneliness and pressure and even bleak physical surroundings set a context for emerging attitudes toward computers, his choice of specialization, and his sense of himself at MIT.

Here's how he describes the atmosphere of his freshman year:

> I was taking too many units at the time .... Because I was on pass/fail, I wanted to get a lot of credits under my belt, and I just drove myself crazy with all these units I was taking .... I mean, that term was pretty nasty overall.

In that year, he took 1.00, which was his first exposure to computers at MIT. He "liked it pretty much," but it was a "time sink." Thinking of the hours he spent in that course in front of a computer screen trying to get the bugs out of his programs, Michel recalls, "Sometimes I got annoyed .... You're just looking at the program and looking at it, and trying to find your error. It can be rewarding when you find your error—all of a sudden the light clicks and ... you say, Oh, thank God!" But when you can't find the error, "that's kind of frustrating." Frustrations were amplified in the physical environment of Building 66:

> By the end of the term, I hated that room. I couldn't stand going there any more, because it's way down in the cellar, and there are no windows in the room. It's just these like four cement walls, it's like five or six long tables, all with computers. Everyone's at their terminal, and you hear all the buzz of the machines, and after a while it gets annoying. It wasn't very aesthetically pleasing, as an environment to work in for, like, four hours at a time. Especially if you worked late at night, which a lot of times happened.

The dreary room late at night was made worse by loneliness—he was "the only civil engineer in my house."

> ... a lot of times I'd go in there alone, just because I didn't have any friends that like were from my house or whatever, with me, taking the course. A lot of times I had to trudge down there really late at night, all alone, and then work on my terminal at the computer, and then would go home. After a while, I started to hate that.

In sophomore year, things began to improve because Michel "started to get a lot of friends." He had declared his Civil Engineering major, and "started making good friends" in the projects in 1.04. Working with other students in his team "helped out a lot."

> People had different ideas ... you know, you had the person to work with and talk with and fool around with, if you wanted to. So it made things more entertaining .... One would type, and the other one would write on a notebook, or just talk ... and maybe we'd take time off, go to a classroom, if we had a

> problem, work it out on a notebook or on a chalkboard or something ... and then go back to the terminal room and start fooling around with it again.

Michel has decided that the computer is a "tool you're going to have to use wherever you go." You'd better learn it, he thinks, "because basically, if you don't use it, you won't pass." He is glad his father bought him an IBM PC, which "saves a lot of time." And in addition, computers are "pretty neat to fool around with." Nevertheless, he limits his involvement, something he attributes to the "kind of person" he is—"That's just me. I get bored." He would never go to course 6.63—"No way!"—although some people "really get engrossed in it." He has heard horror stories about such courses. And he would never "go to the computer room and, like, work on a random problem ... or try to create a game, or something. That's just not me. I'd rather spend my time doing other things."

He describes his temper when he becomes frustrated with debugging, "I get pissed off .... I usually slam my fist on the desk, get up and walk away ...," and has learned to move away from the problem to get some distance. Michel feels under a lot of the pressure from the work load, but a lot of it "from within me ...; I put a lot of pressure on myself." He tries to put balance into his life:

> You want to not only get good grades, you want ... like I want to play sports, which I'll do, and go to parties and meet people, have a good time, and try to get good grades. And you just have to balance the whole thing.

Michel values GROWLTIGER because it's a time-saver, and a source of new insights, recounting his own version of the by now familiar story of an encounter with a surprising result that made him see the behavior of structures in a new way. Michel seems to have attained a satisfactory way of living at MIT. He has learned to use, but also to limit, the pressures of workload and the internal pressures he exerts on himself. He has learned to keep his frustrations and anger within manageable limits. His choice of Civil Engineering—and with it, a way of using and relating to computers epitomized by GROWLTIGER—appears central to his solution. He believes his solution keeps at bay the horror stories and endless frustrations he believes he would encounter in Course 6, allows him to enjoy the solidarity and functional support of a group of friends and to keep the whole of his life "in balance." His life in Civil Engineering and his relationship with the computer fit the "kind of person" he now believes himself to be.

The stories of these three students are tied together by strong unifying themes. All of them chose Civil Engineering as a desirable alternative to Course

6. The "detailed," "abstract," "number-crunching," "theoretical," "miniscule" tasks through which they were introduced to programming lead them to see programming as the province of a particular kind of personality, one opposite to their own. And, in contrast, they come to see certain qualities in Civil Engineering—reality, practical knowledge, relevance to everyday life, large scale, design—that they identify with their own personalities. It would not be too much to say that, in the process by which they choose their specializations, they construct both a typology of fields of study, personalities, and ways of being involved with the computer. It is clear that courses 6.001 and 1.00 introduce students to computers in a way that caused these three to see computers as being for a certain type of person. This may not be the only way to present the material. GROWLTIGER presents itself as accessible to a wider range of "types."

The things these three students like about GROWLTIGER are of a piece with their views of Civil Engineering. It stands for them as a prime example, perhaps the first example they have encountered, of the use of the computer proper to their chosen field—one that subordinates the demanding and potentially infuriating computer to the constraints of practical engineering work. It epitomizes, depending on the student, the qualities of design, creativity and practical knowledge. And it is a time-saver in an environment where the carefully calculated use of time has acquired the status of folk wisdom.

We should also note that the choice of Civil Engineering is, for these students, a way to survive, and even to flourish, in the "fast-paced," "high-strung," "competitive," potentially isolating, and generally pressured environment of MIT. For at least two of the students, moreover, the pressures they seek to manage come not only from the external environment but from within themselves. It is their own compulsion to solve problems, see things through, and do well, that they strive to keep in check. It is not only their vision of Electrical Engineering, but their vision of *themselves* plugging away in Course 6, that frightens and disturbs them.

Part IV: Conclusion

What is the meaning of what we found? And what implications can we draw for policy and practice?

The Meaning of What We Found

* Student Response to the Programs

Expectations for MACAVITY have been disappointed, expectations for GROWLTIGER have been exceeded.

In contrast to the high hopes for intelligent tutors expressed by Logcher, Connor, and Slater, student reactions to MACAVITY have been indifferent or negative, and the consensus among the developers now appears to be that it is perhaps better suited to "slower" students. Is this due to defects in the early versions of MACAVITY? Does it have to do with pitching the level of the program too low? Or is the fundamental educational strategy of an intelligent tutor inappropriate to undergraduates who are (a) very smart and (b) resistant to a program where effectiveness depends on predicting and controlling their behavior?

What shall we make of the finding that students' reactions to GROWLTIGER have been overwhelmingly positive? And that this success goes beyond the program's instrumental effectiveness as an analytic tool that removes the need for tedious calculations?

Students' reaction to GROWLTIGER has been most positive when they have used the program in relatively open-ended design projects that have the additional advantage of giving them the feeling of working on complex "real world" structures. Given the appeal of Civil Engineering for most of the students we interviewed, GROWLTIGER seems to embody what they came to Civil Engineering for. It satisfies a thirst for involvement in design in a department that seems to them to emphasize analysis over design, and for involvement in the actual world of large-scale building in contrast to the "simplicity" and "artificiality" of most of the problems they encounter in their elementary courses. It also provides a way of using the computer that appeals to students whose preferred learning style is experimental, "tinkering," and interactive—a style we have

called "bricoleur." Students whose previous experience with computers suggested to them that computers are made for the meticulous and highly organized "top-down" style of the "planners" here found a computational medium they could relate to.

For a large number of students—by no means confined to those we have termed "programmers"—experience with GROWLTIGER provided a new way of grasping and testing the theories to which they had been exposed in lectures. It gave them, through visualization, multiple iterations of design/analysis, and surprises, a new feel for the behavior of structures. Taking the students' reactions to the two programs together suggests that students react negatively to a program that tries to control them, and seek to assert their independence of it by "beating the system." Students react positively to a program that gives them "constrained freedom," amplifying (through its capacity for automatic calculation) their ability to invent and evaluate their own experiments and to build their intuitions about a limited set of phenomena.

* The Development Process and Some of its Consequences

In general, the development of the CATS programs conforms to the following pattern: the programs' champions take responsibility for developing the programs and promoting their use in the department; other faculty members distance themselves from the programs, ignoring them or making marginal use of them.

Most of the faculty members who have used CATS programs in their courses have assigned them to the other side of the "educational divide" between lectures and labs. They have not learned to use the programs themselves, relegate them to their teaching assistants, and exempt themselves from the development process.

Faculty feel increased motivation to distance themselves because the cost of involvement is perceived as very high. All faculty members we interviewed are alienated by what they perceive as an *unstable* computer environment. In greater or lesser degree, they are disturbed by the disruptions produced by continual changes in hardware, operating system, and software; by the loss of control of their own computer setting; and by changes in the "rules of the game." With each shift in the computer environment, faculty seem to be asking themselves, "Is it worth getting involved?" For those who do not accept involvement in the educational use of computers as an article of faith, the price usually seems too high. Faculty's distance from the CATS programs reflects a more general pattern of distance from students, especially undergraduates. The department's incentive system leads most faculty to limit their involvement in teaching, especially the

more labor-intensive teaching of labs and projects. When students pick up these signals, most respond by staying at arms' length from faculty.

* Reactions to Perceptions of External Control

A significant number of faculty in the department see the CATS program (and more generally, Project Athena) as a form of external control. They resent being told, directly or indirectly, what to do. Several faculty members expressed their displeasure at being deprived of control over their own computer environment, at being physically separated from the computers used by their students, at being required to use Athena's hardware or operating systems when they believe that ordinary PCs would do as well, at being asked to make what they consider artificial boundaries between the use of computers for graduate and undergraduate teaching, or between the use of computers for teaching and for research.

It is ironic that Athena's champions in Civil Engineering advocate the CATS program as a way of achieving autonomy for the department. They see CATS as promising autonomy from externally generated programs. They are apparently unaware of the extent to which their own enterprise is perceived by other faculty as a threat to autonomy. The result here is that many faculty members interested in the educational use of computers have chosen to go their own way, frequently making use of systems more primitive than Athena work stations. They do so because they believe these systems are adequate for their purposes, or because the software they have developed and are most comfortable with does not lend itself to the Athena system, or simply because they prefer to use systems over which they have direct control.

* Consequences for Development and Diffusion of the CATS Programs

The trends described above have several important consequences for the educational use of computers in the department.

The integrity of the development of the CATS programs is undermined.

Most faculty members who have used CATS programs in their courses do not see themselves as responsible for solving problems of implementation or giving feedback to the developers. Students pick up on faculty attitudes and start to see the CATS programs as "forced" or "add ons" to their courses.

The diffusion of CATS programs is restricted.

MACAVITY has had very little use in the department; GROWLTIGER, far less than its potential. At the undergraduate level, GEPSE has been used to date only in Professor Einstein's geotechnical tutor, and is not used outside of the Constructed Facilities Division.

The programs' developers, aware of these patterns, tend to attribute them to faculty's "lack of understanding" of the programs and their underlying rationale, or to "lack of communication in the department." We suggest, however, that the faculty's distancing and its causes help to explain these problems of understanding and communication.

The pattern of use is fragmented.

Because many faculty members have chosen to go their own way, the overall pattern of development of educational uses of computers in the department is fragmented. From a more positive point of view, one might say it is "decentralized," but this has happened without the establishment of the infrastructure and patterns of communication that make decentralized systems function most effectively. Decentralization without communication is fragmentation.

Two additional trends are closely related to those described above, though they cannot be wholly attributed to the fragmented CATS development process.

The configuration of computers in the department is neither centralized nor "coherent" in the sense intended by Athena's proponents.

Three different operating systems are currently in use: BMS, UNIX and DOS. The department's ten microvax computers use VMS; its five Athena workstations use UNIX. The PCs currently employed by various faculty members in research or teaching use DOS.

Faculty members differ in their views of the current pattern of diversity. Some of them deplore it, and wish for a state of affairs in which, as one professor put it, "everyone talked to everyone else and you could use one set-up for everything." Others see the current configuration of computers as satisfactory, or regard it as the inevitable consequence of the conditions under which funding decisions have been made.

There is a spectrum of views among faculty about the effectiveness and potential of the CATS programs, and about the educational potential of computers in general.

The champions of the CATS programs believe that Project Athena has been a success in the department, programs are getting used in courses, and the de-

partmental computer environment has changed for the better. But several senior professors are radical critics of Athena in the department.

They believe that the CATS programs have produced nothing fundamentally new, and that their developers have failed to address fundamental research questions: how engineers think, how students learn to think like engineers, and how computers might help them do so.

Those faculty members who have made marginal use of CATS programs in their courses hold intermediate views, ranging from irritation over the programs' bugs and instability, to positive expectations for the programs' eventual performance.

Cutting across the trends in faculty attitudes described above are a few notable exceptions. These exceptions are the faculty members who see the educational potential of the CATS programs—GROWLTIGER, in particular—and see themselves as contributors to the further development of educational uses of the computer. Professor Kausel stands out in this group as a model of skeptical criticism combined with a collaborative approach to further development.

These exceptions, coupled with the disposition on the part of a number of faculty members to raise fundamental research questions about the development of educational software, represent a significant effect of Athena in the department and a resource for its further use and development.

* How Educational Issues Related to the CATS Programs are Addressed

In spite of Connor's comment that "education is the watchword" in CATS development, the program champions launched into the technical side of software development. They regarded the assumptions about learning that stood behind them as obvious. Among their assumptions: there is "a right way" to solve problems and this is mathematical representation; the best, or only, route to learning is to begin with "fundamental concepts" rather than the intuitions that students bring to problems; students will use tutors according to the designers' intentions. Now, there are indications that some of them have second thoughts. For example, as a result of conversations with other faculty both in the department and in the School of Engineering, Professor Connor has evinced a new interest in the question of "how engineers think" as a research problem.

Feedback from students' use of the programs is a potential stimulus for reexamination of their underlying assumptions. In fact, the program developers have had feedback from students' uses of MACAVITY and GROWLTIGER. A striking example is Professor Slater's role as a provider of assistance and advice

which has put him in close contact with many students. His system of "bug reports" has provided a great deal of information about the programs' technical difficulties.

But much of the feedback tends to focus on technical issues or to be interpreted along these lines. One of the research assistants involved in GROWL-TIGER's development commented that he never actually observes students using the program, or feels the need to do so, but relies on his own experience with it, or on his friends' comments, or on "bug reports." Another research assistant described his worries about the heavy emphasis on technical issues in discussions of student feedback:

> The Athena meetings are very technical. Nobody asks the questions about education.
>
> Although we got feedback from the students, we actually can't tell whether it made a difference to them. Did they like to play with it? What did they learn in class and what did they learn from the program? .... We spend much more time during the design on working on the technical issues, like the interfaces between the machines [used in teaching] .... The culture at MIT is like that. We don't go out and ask consultants on education to help us. It bothers me. Every time we come to a point where we can add a new part to the program, we always have a choice, and it always brings you back to issues like, "Why am I doing this? What is the domain? What do we want to teach?"

And this student pointed out that when he and Slater met with students about MACAVITY, they "got very much involved with the technical issues, like what and where the bugs were, why did you ask us to hit that key instead of this one."

This technical focus, coupled with the developer's natural tendency to read feedback in a positive way, may help to explain the discrepancy between Slater's reading of student reactions to MACAVITY and the students' comments gathered in our study. In sum, the programs' developers tend not to attend to the sort of student experiences that might cause them to question their assumptions about engineering knowledge and education.

* The Educational Research Trap

To the extent that the programs' developers become more interested in questioning their educational assumptions, however, they come up against another set of difficulties. To learn more about the fit between educational assumptions and student experiences, the developers would have to conduct a kind of research—for example, qualitative research into students' experiences with the programs—that

does not lie comfortably within their present range of competences. Furthermore, other faculty members in the department are not equipped to help the developers carry out such research or to evaluate their research results. And in any case, it is by no means clear that such research would be considered an appropriate basis for a young faculty member's advancement in the department. In several of our our interviews with senior faculty members, this point received explicit emphasis. So a young faculty member who wanted to embark on this sort of research would be likely to see it, in career terms, as a dead end—at best, as something to be done "on the side."

These issues are illustrated by the CATS developers' current plan to run a large-scale test of MACAVITY and GROWLTIGER at Texas A&M University. The developers have several reasons for wanting to carry out this experiment. Texas would provide them with the basis for a statistically significant sample—hundreds of students as against the thirty or so accessible to them at MIT. In Texas, where large parallel classes are given on a yearly basis and funds are insufficient to provide every student with access to a computer, control groups would be "naturally" provided; whereas at MIT, the Committee on Use of Humans as Experimental Subjects would not permit the differential treatment of groups of students. Finally, the developers believe that in the student population at MIT, "brightness" has a "camel-back" distribution, whereas in Texas, its distribution is "bell-shaped." This leads them to see the Texas context as more appropriate to MACAVITY's use as a remedial tutor.

The plan for the Texas experiment incorporates a particular conception of evaluative research, one that stresses quantitative measurement, statistical significance, and experimental controls. This experiment may yield useful information. We doubt, however, that it will reveal such important qualitative aspects of student experience with the use of the programs as we have illustrated in earlier sections of this report. In any case, not only will the apparently "scientific" research strategy envisioned for the Texas experiment raise its own difficulties and demands for research competence, but it sidesteps what we believe to be central findings from Athena studies at MIT: Athena reveals educational computing to be deeply intricated in the social world of the departmental and disciplinary culture in which it is embedded; different people make the computer their own in their own way, and understanding this "personal appropriation" demands close attention to the lived experience of educational computing; and the championing of computer systems for educational use, especially without widespread faculty involvement, can contribute to faculty's alienation and distancing.

* The Department's Incentive System

Most of the senior and junior faculty members we interviewed made reference to the fact that the incentive system operating in the department works against serious investment in studying how students learn engineering or how educational software might help them to do so.

One senior faculty member commented, for example, that "senior faculty do not have the time for [this sort of research], given their need to hustle research funds, and junior faculty would be steered away from it in the interests of their advancement in the department." Another senior faculty member observed that the development of educational uses of the computer requires a long-term career commitment; some younger faculty might be interested in making such a commitment if funding for it were available; but there are no such funds.

The comments of younger faculty members' indicate that they get the message. One of them said that "Artificial Intelligence or educational research won't get you tenure; nor will it get you a position in another university if you fail to get tenure at MIT." Another spoke of the "insane environment at MIT" where "you get no brownie points for teaching or work on Athena."

The same incentive system that militates against faculty involvement in the research questions in education also helps to account for faculty's tendency to keep their distance from the time-consuming teaching of labs or projects where students are most likely to be using educational software. Several faculty members, senior and junior, said that they simply don't have the time to work with students in labs or projects—"three hours an afternoon, three afternoons a week," as one of them noted—when they need to be "out chasing dollars for research."

Some Implications for Policy and Practice

A number of factors work against the CATS program achieving its educational objectives. The isolation of the development process has kept other faculty members out of the business of providing useful feedback on the educational effectiveness of the programs or contribute to their further development. The UNIX Athena system has reinforced the tendency for many faculty to go their own way. The result: the diffusion of the CATS programs has been limited and there has been relatively little collective involvement of faculty in the further development of educational software. In the CATS development process, technical issues have tended to predominate over educational ones. And although a number of faculty members now believe that Project Athena's objectives in the department call

for long-term investment in the study of fundamental questions about the nature of engineering practice and engineering education, both the department's present configuration of research competences and its system of incentives militate against such an investment.

In spite of these impediments to its successful evolution in the department, the CATS enterprise has already had significant educational impact. The success of GROWLTIGER is impressive in its own right. Moreover, some students' experience with GROWLTIGER suggests exciting possibilities for the further evolution and use of educational software.

In spite of the general pattern of faculty keeping a distance from the CATS initiative and from students' experience with computers in labs and projects, several faculty members have collaborated with CATS developers in extending or adapting the basic programs to new educational applications, and some others have plans or hopes to do so. At least one faculty member has seriously interested himself in GROWLTIGER and thought critically and constructively about the future uses of programs like it. And, as we have already noted, the Athena experience has prompted a number of Civil Engineering faculty members, including at least one of the CATS developers, to become seriously interested in fundamental questions associated with the development of new uses of the computer in engineering practice and education.

All of these things must be counted as significant effects of Project Athena. In addition to the achievement they suggest of themselves, moreover, they suggest ways in which the CATS enterprise might be more productively pursued in the future, should the department's faculty members, or a sub-set of them, decide they wish to do so:

1. Explore the directions suggested by students' experience with the programs to date. Here, it would be especially fruitful to consider GROWLTIGER's effectiveness in students' relatively open-ended design projects, and its use as a vehicle for testing structural theory and connecting it to feel for structural behavior.
2. Broaden the scope of CATS to include all faculty-initiated projects in the educational use of computers. Within this broader compass, open up the boundaries to include the several kinds of hardware, operating system and software now employed by faculty members in the department. Make the boundaries between research and educational uses of computers more permeable. Rely on faculty learning to guide the degree to which faculty members should converge on a shared choice of hardware, operating system and software.

3. At least in the further deployment of computers (if not in the redeployment of existing computers), allow faculty members to remain physically promixate to the computers they use for educational and research purposes.
4. Work toward a collaborative and consultative development process in which other faculty members share control with the CATS program developers.
5. Decrease the distance between faculty members and student experience with the educational use of computers by encouraging faculty members to learn to use the CATS programs and cross the "educational divide" between lecture and lab or project. Encourage faculty self-studies of students' use of educational software.
6. Explore how the department might legitimize the development of skills for research into a set of fundamental research questions that stand behind the field of computers in education—how engineers think, how students learn engineering, how computers help them (or might help) do so.

It is worth considering how a development process of this kind might become part of the department's strategies for increasing its share of MIT undergraduate majors and staking out areas of front-running research in its field.

We should note that, in our interviews with faculty members, we found several different views of the strategy most likely to be effective for the department. These can be described, in rather oversimplified terms, as follows:

- infuse Civil Engineering with the new research findings and competences of sciences like chemistry and biology;
- return to the traditional strengths of Civil Engineering—knowledge of the appropriate uses of materials and structures;
- focus on preliminary engineering design, away from the sort of detailed design that will be relegated to the computer;
- focus on the integrated design of large-scale engineering systems;
- restructure the field in terms of knowledge engineering—the use of computers for the development of knowledge-based systems;
- focus on the management of engineering projects; and
- focus on the development of expert, or "smart," systems through an AI approach to the study of how skilled engineers really think.

Although some faculty members advocated one of these views, it was not uncommon to find faculty members who held several of them at once.

On the basis of our analysis of interviews with a limited sample of undergraduates, some of these departmental strategies seem likely to work better than oth-

ers in terms of drawing students to Civil Engineering.

The majority of students in our sample were attracted to the department because of their view of the traditional values of Civil Engineering which they saw as concerned with large-scale, real-world, highly visible, socially relevant projects of design, building, and systems implementation. While these students tended to recognize the importance of mastering the "basics" of structural analysis, their interest was primarily captured by the idea of *design*. They were mainly responsive to the use of computers as tools for design and as vehicles for learning how to design—as exemplified by GROWLTIGER. Their reactions to the department's introductory computer course, in contrast to 6.001, could be taken as a reinforcement of the importance of that course as a pathway to the department. For the MIT undergraduates represented by this group, a departmental focus on engineering design is likely to serve as a powerful attraction.

Project Athena at MIT: Computing in the Department of Chemistry

by Sherry Turkle and Brenda Nielsen

May 1988

Introduction

Although computers have had a long history in the Chemistry Department's research activity, the history of their use in undergraduate education is short. Essentially, it is the story of a small number of faculty. One of them, Dr. Dagmar Ringe, director of the undergraduate laboratory and lecturer in the department, became involved in computers and education on her own initiative prior to the formation of Project Athena.

By 1981, Dr. Ringe was convinced that students would be better able to grasp the significance and underlying theory in laboratory experiments if computers performed the tedious calculations essential to the analysis of experimental data. Dr. Ringe arranged to have several Commodore PET computers available in the undergraduate laboratory. These machines had limited capabilities, but were sufficient for the simple calculations they were asked to perform. Students entered experimental data and received numerical outputs. Until 1983, this pilot project was the only educational use of computers within the Chemistry Department although there was a widespread feeling among faculty that there should be more. Dr. Ringe gave voice to their starting assumption, "Computers are almost everywhere in chemistry research." Her colleague Professor Jeffrey Steinfeld stressed the practical imperative as well: "In the laboratories that students will work in upon graduation, everything is computer based, everything is microprocessor based."

The Chemistry faculty has long agreed that students need to be computer literate. Still controversial is what that actually means. Is it sufficient to provide students with opportunities to use existing programs or should students have programming expertise as well? Professor Keith Nelson describes computer literacy as the ability to use software, but stops short of saying that programming skills are necessary. His analogy is with the way chemists use other kinds of equipment. "I didn't build the AMR spectrometer the students use in the lab. And I didn't put together the electronics for the IR spectrometer either. And I don't know how to. We bought these things .... For most users commercial software packages are pretty adequate."

Others disagree. Particularly among students there is strong feeling that programming skills are critical to the scientist if he or she is to feel completely at home in the contemporary research environment. Without such skills, researchers

on the effects of computer use in the laboratory. The second, MSGD (Molecular Structure Graphic Display) has had a far more troubled history. We discuss it in a second section where its difficulties illustrate roadblocks to success in educational computing that need to be dealt with to increase the chances for better results in the future.

The study of Athena in Chemistry is dominated by the fact that few faculty members have sought a direct role. Central participants have tended to be recruited by administrators. It is important that the reticent faculty include members of the department who have considerable professional interest in educational computing. Here we try to understand what stood between them and participation in Athena. This is are relevant to the Chemistry Department's continuing efforts to use computers in its undergraduate program and Athena's larger attempt to respond to the needs of the MIT community, especially in the School of Science.

This report is based on interviews with seven faculty and staff members, eleven students and four teaching assistants in Chemistry. The interviews varied in length from one to three hours. Faculty and TAs were asked to comment on the use of the computer in the department, on their evaluation of its impact on students, and on their experience with Project Athena. Our interviews included faculty who chose not to involve themselves in Athena or who chose to discontinue their involvement. The eleven students interviewed (six men and five women) represent half of the students enrolled in the 1986 Advanced Chemistry Lab. They were asked to reflect on their experiences with PEAKFINDER, other computer experiences, their career plans, and to share more general thoughts about their education in Chemistry.

In addition to interviews, we spent approximately twenty hours observing the experiment in which PEAKFINDER is used—the lab work as well as the computer-aided data analysis. We also attended public meetings and presentations that related to computer use in the department.

Part I. Athena Comes to Chemistry

The Advanced Chemistry Laboratory

Advanced Chemistry Laboratory, 5.33, is the third and most demanding course in the Chemistry Department's integrated laboratory sequence. It focuses

other values needed to calculate bond lengths and to generate a model spectrum based on values specified by the student.

The value of the lab, as Nelson sees it, is that students can look at the recorded spectrum of molecules and "can ultimately relate the position and intensity of the peaks on the spectral graph to fundamental quantum mechanical models of how the molecule works." In other words, the IR experiment gives students an opportunity to verify quantum theory for themselves.

Professor Nelson recalls that prior to the introduction of the computer into the experiment, students spent much of their time in tedious manual data reduction. Students worked from a long chart of spectral data, measuring distances between peaks representing maximum absorption with a plastic printer's ruler and assigning position to the peaks. The analysis of the spectrum took "many days of number crunching and staring at lines. The process was repetitive, mechanical, and laborious." And the process itself did not necessarily lead to greater insight into quantum mechanical theory.

Students who had known both ways of doing things, without the computer and with it, agreed with his assessment.

> Spectroscopy was a pain to do. It took 15 hours of students' time. You didn't learn anything by reading graph paper or typing numbers into your calculator. It was prone to error and boring. it really turned people off the lab .... That had been happening for years.

Student response to the use of computers in the experiment was overwhelmingly positive. Teaching assistants were well trained to help students with the new equipment; they had made a special point to identify bugs in the software and were able to guide students through trouble spots. The teaching assistants remarked that things went a lot smoother for students who had PCs of their own.

> They know how to turn on the thing and format the disk. If the machine crashes they know what to do, there's no problem. The people who don't have experience are petrified and mortified if something goes wrong. The people with experience are very cavalier; they just go in. The computer doesn't bother them because they know they can't hurt it. The people who don't know that ... wonder, "Should I really hit this key?" They are worried about it .... They seem to learn pretty fast though.

Students did learn fast (several said that their difficulties with equipment centered not on the computer but on the vacuum pump used to collect the gas sample). PEAKFINDER decreased the number of manual calculations and the cursor

Essential to Nelson's way of thinking about the computers in Advanced Lab is that they are good for some things and not for others. They are useful tools for relieving tedious tasks, but can play only a limited role in making students creative chemists.

> PEAKFINDER is an efficiency tool, basically. It does certainly free up time for students ..., but it isn't trying to give them a deeper insight into the material. At this stage at least, that's really not what is happening.

Nelson is not alone. In general, the Chemistry faculty think of their field as divided between what they see as the routine (working out the details of an experiment, collecting accurate data, running numerical analyses) and what is exciting. In this second category are designing an approach to a problem, challenging assumptions, exercising intuition, and developing insights about theory.

In teaching, faculty consciously focus on what they consider most exciting in order to convey a sense of what chemistry is "really about," the "puzzle and the mystery," a sense they fear may be dulled in students who have to spend a lot of time on the routine. And they see the computer is part of that routine; it doesn't enter into their plans for how to communicate "things that matter." Nelson thinks that these are communicated "when students sit down with a book, a pencil, and paper." Professor Gregory Petsko, a biochemist, talks about how he tries to communicate "excitement for ideas and theories" in his lectures where the computer cannot help him, except perhaps, as a drawing medium to make up for the fact that in his words, "I can't draw worth a damn."[2]

Nelson and Petsko, like most of their colleagues, can see the computer as a calculating tool or a better replacement for chalk, but they don't expect the computer per se* to cause qualitative changes in students' grasp of the material. "The IR experiment is the same as it was before," Nelson maintains. "Without the computer, students would still have to analyze the spectrum. It would take them many days, that's all."

But we have noted that although some students agreed with Nelson that the practical help the computer brings has not of itself brought any new depth of insight, the majority see the use of the computer as making possible a different and deeper understanding of the material. "It's not so much that the computer saves time, but that with the computer I can do things I couldn't do otherwise." "The

[2] Even if the computer could help him, it is not clear that Petsko would want it to. Classroom teaching is to generate excitement; "I want to keep that function for myself."

Dr. Ringe describes what she thinks has changed for the students: by giving them precise data to work with, computers put them in an active position. more active position. With precise data, students can verify or challenge theory. The imprecise data that they used to have did not give them that kind of confidence. Students trust the accuracy of the new data and when it matches theory, the theory seems "true," "believable," more personal.

One of the things students discover when they use precise data is that the phenomena described by the quantum theory are complex. Rachel gives an example:

> The separation between peaks are all supposed to be the same, but they are not because the molecule isn't perfect. It isn't as simple as the model is. The model that you think of has little balls and springs that attach them, but that's really simple and it doesn't take into account the fact that the way the molecule vibrates isn't exactly harmonic.

She goes on to describe how the PEAKFINDER program helps her to deal with those variations and to generalize from them to fit the theory.

> If you rotate something really fast it tries to fly out. The bonds between its parts get distorted. That will affect your theoretical expression. So, you have formulas in the computer that account for the unequal spacing. It takes your math expressions including the corrections for each of the lines and it calculates the difference in spacing for all of the pairs.
>
> That's what the straight line plot tells you, how well the expression that has the correction explains the spacing between the lines. By seeing the line plot you know that the theory is supported by your data.

In other words, students come to understand that the connection between theory and data holds on a general level, in spite of the idiosyncratic characteristics of individual molecules.

Students also express enthusiasm about the actual interaction with the computer. ("It was neat to see!" "You could actually see it draw the spectra out. It was incredible.") They say it generates a surprise and excitement that keeps them "at it," that keeps them "going." In this lab, the immediacy of the machine's response and the fact that it concretizes theory is central to the machine's "holding power." There is, too, another source of holding power. This is the way the computer leaves open the possibility for its personal appropriation. Even in an application as "straightforward" as the collection and analysis of spectroscopic data, different students make the computer their own in their own way.

a predetermined plan. Chemistry and the computer are valued because of their ability to "surprise."

The fact that people use computers in diverse ways has important implications for educational computing, a field which often takes for granted a uniformity in how software will be used. For purposes of our presentation here, we simplify the wide range in styles of computer use and limit ourselves to a discussion of what we call "planners" and "bricoleurs." Planners tend to reason with theoretical abstractions and pre-planned strategies. Bricoleurs tend to explore the world by playing with the imprecisions of "wet" chemistry. Theirs is a process closer to "tinkering" than planning. Planners and bricoleurs differ not only in their style of organizing work, but also relate differently to whether the computer needs to be "transparent" to them, the degree to which they need to understand what is going on inside the machine or inside the program they are using.

Beth, a typical bricoleur, is willing to let the computer do its job without requiring transparent understanding at every step. She is willing to have the computer relieve her of the burden of step-by-step calculation so she can better concentrate on "learning the molecules." For her, the lines of the spectral graphs were like "seeing the molecule moving ...; the first time we really saw what was happening." She begins by trusting the computer—it frees her to explore. Her process of discovery is through impressionistic glimpses of the phenomena at hand. And then, after profiting from the discovery environment the computer offers, Beth tries to make it somewhat more transparent. For "without understanding," she says,"you could just make pretty pictures."[3]

Planners tend to strive toward a more transparent relationship with the machine. Mark reflects this characterization. He cares most of all about precision. He loves computers for their accuracy, but they can only reassure him about accuracy if he is working with a program that he can follow step by step. When he writes his own software, he can reliably get to this feeling, but he realizes that "the overhead is too high." He has no choice except to use prewritten applications programs. And these pose problems. Mark liked PEAKFINDER because he

[3] The kind of transparency we are talking about here, being able to follow the program step by step, is different from that referred to when the faculty expresses concern that computer use can obscure the process of analysis. In her very first exploration of laboratory computer use with the Commodore PETS, Dr. Ringe discovered that many students did not understand the data they were entering and many did not understand what specific calculations were being done by the computer, "they were simply entering a set of numbers, getting out an analysis, a number for which they had no clue if it was reasonable or not. They didn't even know whether it was a straight line or a curve." At that point she intervened: the programs were modified so that students had to specify the analytic steps that the computer would carry out.

Part II. Roadblocks and Possibilities

The PEAKFINDER program for the Advanced Chemistry Laboratory was considered a success. The program was developed smoothly and was introduced as a matter of course. But PEAKFINDER was a relatively straightforward piece of software. One very talented undergraduate was able to write it as a summer project. By contrast, MSGD illustrates that when educational software is more technically ambitious, things are not so simple. Its troubled history serves to illustrate a set of themes that have contributed to the reticence of Chemistry faculty to get involved with Athena.

The Molecular Structure Graphic Display Program

Professor Petsko has long used VAX computers to generate visual displays of large, complex molecular structures. The vast quantities of data needed to display and manipulate these molecules require the use of large computers. However, Petsko is also interested in the interaction of smaller molecules with the large ones, and has found that the VAX is "too much computer power for that kind of operation. That kind of construction and manipulation can be done in a more natural way on a smaller computer." With that in mind, Petsko got in touch with programmers at a software company [eventually to become Polygen] to discuss the development of a software package that enabled one to draw and manipulate small molecules. This, in turn, would provide input for the interaction of the small with the large molecules that were being studied on the VAX. Petsko's contact with the software company occurred at the time that Project Athena was getting started.

Dr. Ringe explained how Petsko's efforts to acquire a modeling program for research purposes grew into the Chemistry Department's second Athena project.

> By a rather circuitous route, we had an offer from some outside people to work together on a modeling program. They had the expertise in programming and graphics, none in chemistry. We had the expertise in chemistry and none in programming and graphics. Putting the two together, we thought we could develop a very nice program for modeling in chemistry.

Dr. Ringe knew that three-dimensional computer generated displays of molecular structures held educational promise. "Students frequently have trouble visualizing the structure of small molecules, understanding how they look in space and how their isomers look." And for many students, the conventional stick and

is difficult for either faculty or students to do "on the side." It is not easy to adapt a program for educational use that was written with altogether different purposes in mind. And it is not easy to get it running on hardware that is limited in relation to this purpose.

Professor Greene had become intrigued with the possibilities of MSGD during an IAP open house. His path of recruitment into Athena was unusual. To date, few members of the Chemistry faculty have been involved in Athena-related projects, and of the participants, Dr. Ringe is the only one who was not recruited by administrative request. The Chemistry faculty has been notably reticent. Now we turn to why, focusing on reasons that have inhibited Athena development beyond the confines of this one department. In this exercise, it helps to keep the history of MSGD in mind. The Chemists feel that it demonstrates the problems they face. Bringing computers into the curriculum is not a simple matter, certainly not an "exercise for the left hand."

Faculty Reticence

Athena began with the assumption that a lack of hardware, facilities, and updated technical information was keeping MIT faculty away from developing ways of including computers in their teaching.[4] But of course, things are more complex. When Professor Kinsey commented on faculty "aloofness and hostility" to Athena, he stressed that in his opinion, it had more to do with an institutional incentives system than lack of information or availability of hardware.

> Given the pace of life in this kind of business, there are a lot of faculty who say, "Don't come and tell me to start teaching some different way. I know how to teach. I've got my lecture notes. I don't have time to do all the things I do now. Don't come and give me some new chore."

There are realistic limits to the resources faculty can devote to innovations in teaching. Kinsey believes faculty divide their time as a function of their research commitments and responsibilities to sponsors. Such commitments leave little time to pursue educational innovation. The incentives within the profession paint faculty into a corner.

[4] "Introduction to Project Athena," The Project Athena Executive Committee, October 1983.

to develop it. Athena's hope that faculty would do it seems unrealistic to the Chemists. It encourages them to stay out until they see a radical change in expectations.

Athena has also supported the idea that students would serve as software developers. The Project Athena Executive Committee took an early optimistic tone about student programmers, referring to MIT students as a "pool of creative talent on which the project will draw heavily."[5] And indeed, the Chemists remark with some irony that "insofar as we have 'quote' developed software, this consisted of finding some smart kid who could write the programs." But they also know the limitations this approach.

In Chemistry, the pool of student talent has turned out to be very small. Incoming Chemistry graduate students are carefully screened for their potential ability to make an Athena contribution, but Professor Steinfeld reflects that "with all we hear about computers in schools in recent years, the number of students with such a background [in programming] is low. Many have some exposure to computers, but in terms of the real depth of understanding that's required, it is really only one or two people." At present, the two Athena-funded teaching assistants are occupied—one writing front-end software for MSGD and fixing bugs in the PEAKFINDER program, and the second testing the user manual being developed for the MSGD package. Until they complete these projects, there is little or no manpower to undertake new ones.

This lack of student resources increases the faculty's sense that Athena's expectations are out of touch with their reality. Even when graduate assistants with the requisite skills are found, the yearly turnover of TAs slows down the development process. Most Athena-funded TAs spend only their first graduate year on the job before moving on to research in their field of interest. Their expertise and specific knowledge of the projects under development is lost.

One of the MSGD TAs is an exception to this rule. He has worked on the project for two years. He feels he is more productive this year [1987] than last because he knows the programs and the hardware. However, by the end of his second year with the Athena project he, too, will join his advisor's research group. "The work I was doing isn't finished, but my research advisor would rather see me down in his lab than up here being a TA."

The Chemistry faculty is highly computer-literate. It is *because* they understand that software development requires specialized skills and large amounts of

[5] "An Introduction to Project Athena," *op cit.*

Chemistry had itself bought equipment that Athena had not been able to provide. The department had repeated scaled down its expectations of things to come. But planning had become next-to-impossible.

> We kept going from one interim solution to another as Athena didn't deliver. We were trying to plan a little further down the line in terms of a certain environment and then that environment never materialized. We were constantly falling back and making progress at the same time.
>
> You expect some things, those never materialize and you have to re-program your efforts. If one had had a little more realistic picture from the start, the whole effort would certainly have proceeded with fewer false starts.

Some Chemistry faculty go so far as to say that educational computing in their department was actually held up by Athena. According to Dr. Ringe,

> Athena ... delayed the computerization of the labs. The faculty knew that "Athena was coming." They didn't know it would take two years to arrive. If it hadn't been for Athena then the department would have done it on its own, and it would have probably been done faster .... Now it's a matter of patience, outwaiting Project Athena.

Dr. Ringe has stayed involved despite all the frustrations, but many faculty members have preferred to wait until MIT can deliver equipment at what they consider to be an intellectually interesting level. This is particularly true of the most technically sophisticated ones. Given what Athena now provides, to them, participation seems a source of potential frustration rather than gratification.

For example, Professor Petsko knows he could use graphics software in his teaching, but feels that doing it with the current hardware would be more trouble than it is worth. With the "three ATs now in the undergraduate lab," he describes it as "hopeless."

> Every single terminal available in the Department should have the proper graphics and, in the ideal world, a large file server networked to it. Those are the configurations we were promised when Athena was set up and they have been horribly slow and ridiculously inefficient in getting the stuff going.
>
> Whether the fault lies with Athena or IBM I don't know and I don't care. They ought to get their act together! Otherwise, this thing will never be more than a pipe dream, and it will never have the educational impact that it could have with proper hardware systems. My god, there have been adequate graphics boards available for five years now!

experience with computers, in particular those that went beyond the faculty's intentions.

By and large, faculty saw PEAKFINDER in practice as they had seen it in theory. They viewed its performance in terms of their intentions—as a tool to increase speed—despite the fact that many students experienced it as far more. This is not surprising given the prevailing faculty tendency to separate "teaching" (the exciting, in lectures) from "learning" (the routine, in labs), and to relegate computers to the latter and to lower-status individuals (primarily TAs).[7] But this situation needs to be remedied to insure that faculty receive adequate communication about the impact of educational computing programs.

Students' report that PEAKFINDER changed their experience of chemistry opens up a domain for further investigation as does the fact that different learners adopted different styles of software use. It is clear, for example, that opaque, "idiot-proof" software will turn many MIT students away from the computer. Developing a range of style-appropriate tools will turn many towards it. The question of how to exploit these effects to maximally enhance education requires further research. This research is certain to have important practical implications.

For example, when Professor Kinsey talked about his style of working through problems, he described a highly visual approach, "drawing little pictures" of the molecule he is working on. Kinsey, in our terms a "bricoleur," looks to the computers of the future for tools that would allow his style of problem solvers to do flexible, visual explorations of their data, looking at "an enormous number of examples, changing a lot of parameters and what-ifs." Our study suggests that computers may facilitate learning by making it easy for different people to use different intellectual styles and by providing a new point of leverage on maximizing their effectiveness.

The Chemistry faculty has responded to the demands on it by favoring a model for the development of educational computing that looks to professional software developers. There is consensus that Athena's original model of using faculty and students as programmers was unrealistic. The department's disappointing experience with MSGD where graduate students tried to "patch up" commercially written software has only reinforced this consensus. It has emerged in a group of dedicated and technically sophisticated educators who feel they have had

[7] Among faculty, Dr. Ringe was something of an exception. Her special role, her continuity in an "intermediary" position as head of the undergraduate laboratory, contributed to her having a somewhat wider view of the potential of educational computation and a more accurate view of what PEAKFINDER could mean for students.

Project Athena at MIT: Computing in the Department of Physics

by Sherry Turkle

May 1988

Introduction

In our study of the educational impact of Project Athena at MIT, the Physics Department was chosen as a case study because of its reputation as a somewhat resistant intellectual milieu. What we found was far more complex: an environment where it was possible to understand serious and significant reticence about educational computing and about the directions taken by the Athena organization. In Physics, Athena "reticence" must be understood in terms of such issues as the intellectual commitments of major figures within the department, philosophical questions raised by computer use, and a clash between the department's collective intellectual personality and the Athena management style.

The Athena experience has led members of the Physics Department to a heightened sensitivity to questions about the substantive effects of educational computing; from the point of view of Physics faculty, computers in the classroom raise issues that go beyond pedagogy to touch on fundamental questions of how the scientist approaches the real.

Background to Resistance

For the past twenty-five years, computers have been an essential element in physics research because of the need to process large amounts of data. FORTRAN, good for writing fast code in a convenient environment, became the language of choice, and deeply embedded in the physics culture. Thus, the sophistication of most physicists about computers tended to be concentrated in a particular kind of computer culture: the data processing culture.

Of course, in recent years, personal, dedicated computers have created possibilities for scientists that have little to do with "number crunching." The new machines allow a speed and interactivity in data collection and analysis that opens up a different world for which physicists had little intellectual preparation.

The domination of FORTRAN and data processing is background to a general hesitancy about personal computers among physicists. At MIT, this general hesitancy was magnified by the fact that a number of distinguished members of the MIT Physics Department saw computers as a problematic path of access to physics. For example, Victor Weisskopf, longtime chairman of the department, now professor emeritus, feels that the computer takes physicists away from

making possible a new class of numerical solutions, French gives voice to the fear that they can have the opposite effect as well: masking physical realities. "In general, students come here innocent of any particular acquaintance with with the real world, the physical world." French regrets that the lack of a required formal laboratory in the Freshman year, allows this situation to continue. So, "if there is anything at all that can be achieved by direct experience, I would want to go for that rather than for the substitute." And computers, by making demonstration and simulation so easy, encourage such "substitutes."

> I think there is a crying need for our students to get some exposure to what it means to make a measurement and arrive at a conclusion, which can be backed up by experimental observation with a stated degree of error. And I think the idea of anything that would simulate that experience and provide a substitute for it, that's one aspect of the uses of computer which I really don't like at all.

The theme of dual effects, the computer revealing and the computer masking, runs through the reaction of physicists to educational computation. Computers are good when they make it clear that the real world is characterized by an irregularity which demands a respect for measurement and its limitation. Computers are bad when they interfere with the most direct appreciation possible of the real. In this context, computer simulation is acceptable when there is nothing else to work with, as in the "invisible" areas of quantum physics. But physicists stress that from an educational point of view, it is always better to get your hands dirty. If simulation becomes too easy, students will turn away from the messy reality to which they owe a first allegiance.

Overview of this Report

There are four main Athena activities within the Physics Department.

- A Freshman Seminar on computers in physics, initiated by Professor Robert Hulsizer, and currently taught by Professors Robert Ledoux and Steven Meyer.
- A set of computer programs used in the experiments in the Junior Laboratory, developed by Professor Martin Deutsch.
- A software project directed by Professor Edwin Taylor to develop interactive simulations in quantum mechanics and relativity.
- A course in computational physics taught by Professor John Negele.

the real world by distancing them from the steps between problem and solution. When colleagues would show him their print outs, he was fond of saying, "When you show me that result, the computer understands the answer, but I don't think *you* understand the answer." Herman Feshbach followed Weisskopf as chairman. With Phil Morse, another influential MIT physicist, Feshbach wrote a classic two volume work on methods of mathematical physics which made an implicit statement about a hierarchy of methods. One of their colleagues put it this way: "If you are really gifted at solving problems in mathematical physics, you might have as a corollary that somebody who has to resort to a computer to solve problems is not as clever as somebody who can solve them with mathematical techniques."

On the other side, MIT Physics has a tradition of supporting innovation in physics education. Out of this commitment came the work of Jerrold Zacharias, principal author of the PSSC Physics Curriculum. In the 1960s. Professor Anthony French of the Physics department worked with Zacharias, as did Judah Schwartz and Edwin Tylor who although not members of the department have contributed to Project Athena activity within it. But Athena was not in a good position to capitalize on the department's tradition of commitment to educational innovation. There was a clash between faculty members' sense of appropriate process and Athena's high degree of bureaucratic centralization. From the faculty's point of view this centralization expressed itself in Athena's tendency to tell faculty what was good for them. The Physics Department is a community of independent, successful, and strong-opinioned scientists who are accustomed to making their own decisions and setting their own educational priorities. As a group, they were offended by Athena's style.

But beyond the issues of style were questions of intellectual substance, anticipated by Weisskopf's skepticism and the implicit hierarchy which gave preference to analytical solutions. The traditional textbook presents problems that can be solved analytically, mathematically. The computer allows even the beginning student to work on problems that can be solved only numerically, by taking measurements and fitting curves. Anthony French put it this way:

> The chief difficulty of teaching elementary physics is that it tends to be presented as a set of absolute truths. The student never sees enough of the provisional, limited nature of that knowledge of the universe. Students don't see the things that go outside these oversimplified models; they are simply not things you can handle in simple terms. But the computer makes that possible.

But although computers may help to expose traditional oversimplifications by

> I, like many of my colleagues, have an instinctive negative reaction to anything that smells at all like something that will regulate or control our activities, because basically our success, when it comes, comes from the fact that we disregard such things.

Athena offended Deutsch with its rules and constraints: "They are going to tie us to mainframes .... They are going to give us all terminals. They are going to tie us to a system which *they* know is the only one to use." For Deutsch, Athena's centralization, terminal rooms, and list of "acceptable" languages, harked back to an old way of doing things, the way of doing things that predated the advent of the personal computer, something that had been very important in his own intellectual life.

Deutsch discovered computers in the 1960s with the PDP-1. For a few years he devoted himself to that machine, writing programs in machine language, getting involved in its internal architecture, "having a good time." Then, for ten years he had nothing to do with computers until he met the Apple 2E: "I bought a 2E and I said, 'That's going to change my life.' And it did. For the first time in my life, I could write."

Styles of Mastery

Deutsch does not write or solve physics problems by making a plan and following it through top to bottom. He prefers to assemble the elements and "sculpt" a solution. The combination of this sculpting style of work and a perfectionistic nature had always made writing painful to the point of near impossibility. Deutsch found it too linear a process for his associative style. The computer presented a new opportunity.

> A person like me can never write anything. By the time I had five sentences down, I'd start correcting them. And by the time I'm through correcting them, I've lost the train of thought. So I used to do things like use a tape recorder, put it at the far end of the room, hang up the microphone, walk over to the other end of the room so I couldn't get to the machine in time to stop it, you see, and start talking. Because once you have a draft, you can work on it. And of course, the word processor solves all that. You can just put it in any old way and then start working on it.

Deutsch's "sculpting" approach is characteristic of the "bricoleur," a style of mastery that eschews planning. Like a paradigmatic bricoleur, Deutsch admits that he "never reads the literature first. I first try to solve the problem." Deutsch

moved directly from word processing to graphics programming and then turned to the question of how to use a system of networked Apples in the administration of the Physics Department. He worked with the secretaries and administrative staff. There was no overall plan. Strategy was "sculpted" as they went along. The idea was to make the program self-correcting and responsive to the participants. It became Deutsch's model for innovation.

From this experience of introducing computers at the "grass roots," Deutsch was left with a sense of a great divide in how computers can enter organizations: in a centralized or in a decentralized way. He equates centralization with oppression and decentralization with liberation. Computers can be on either side. If control is decentralized, the computer can empower: "The poor bank teller who used to only be able to say that the manager was out to lunch and only he had access to information, can now manipulate data on the screen. And she can suddenly have the feeling that she is the bank." If control is centralized, the computer can alienate; people end up with less information rather than more. For Deutsch, the computer can also create another kind of alienation, an epistemological alienation if the machine is presented as a "black box" to the user. In his view, opacity leads to alienation, both intellectual and political.

In contrast with his colleagues, whose experiences with computers had been linked to batch processing and FORTRAN, Deutsch's view of how computers might be involved in education was molded by the experience of personal computation. He saw the machines as entering the researcher's intellectual space. He saw them as personal tools which could be customized for learning. He had used them to get into a new relationship with words and saw little reason why they couldn't be used to get into a new relationship with scientific data. But this would not follow from the old model of feeding data in and using a FORTRAN program to get it out. It would follow from a new model that emphasized personal use.

Deutsch began to approach his colleagues about using computers as personal tools in physics. From his point of view, he was met with indifference and hostility because they did not know what he was talking about. "Everybody knew what a computer was. It was in a computer center where you submit your stuff in the evening and pick it up the next day." And I said, "Look, this is really going to change. We're really going to have new things." Deutsch created an ad-hoc committee on computers to "stir the pot," to create an awareness of the new possibilities.

It was at this point that Athena became an actor. Deutsch thought its ideas

about networks were interesting, but in the short run, impractical. More exciting was an opportunity to bring computers into students work spaces, mental as well as physical, just as they had entered his. Deutsch's idea was simple: introduce personal computers into the Junior Lab.

The Junior Lab

Deutsch wrote a series of data collection and analysis programs for Junior Lab. Although the programs were not very different from commercially available software, doing them himself allowed him to customize the software, and of course, following his computational aesthetic, to make this software transparent to the user.

Deutsch begins with the assumption that the most dangerous thing in a laboratory is a "black box"—a piece of equipment or a procedure that students do not fully understand. Ideally, each "apparatus should be simple enough so that the student can open it and see what's inside." For practical reasons, students can't design their instruments for themselves, but his ideal is that students should *feel that they could have*. Deutsch's educational philosophy is based on his understanding of his own learning process: good learning requires intellectual ownership. For Deutsch, you can't have intellectual ownership if you are not working in a transparent intellectual environment.

As Deutsch sees it, he has been fighting a battle against the black box all his life, "a rear-guard action" because, "as techniques become established, they naturally become black boxes." But he says, "it's worth fighting at every stage, because wherever you are there is a lot to be learned" if you keep the box open. And you will miss it if you allow the box to be closed. He sees the computer as a new weapon in his battle for transparent understanding because it is incomparable as a means of forcing the steps in a process to be made explicit. For his first efforts in using transparent computation in the laboratory, he looked for experiments that were "thin on the physics." He did not want complex theory to stand in the way of students getting close to the experiment, the "key to what separates a physicist from an engineer .... What the engineers do most of the time, when they say they perform an experiment, what they are really doing is performing a measurement. That is not quite the same thing."

A first simple experiment using the particle spectrometer was designed to increase students sensitivity to errors. When a source of electrons is placed first in a magnetic and then an electric field, it is possible to determine its specific

charge. Deutsch observed that before the computer became part of the data analysis in this experiment, students were not likely to take account of errors. "They had very little feeling for what an error meant." Deutsch uses the computer to force a change. *His program will not accept a data point without the specification of an error factor.* If students insist, Deutsch will show them how to remove this feature from his program. But to do so, they "have to take affirmative action. The default is always to put an error in."

A second experiment, known as the Stern-Gerlach experiment, has a similar effect of bringing students closer to the data. As an electron travels around an atomic nucleus, a magnetic field is set up. Classical physics predicts that, if influenced by another magnetic field, the orientation of the electron will be deflected. One expects a continuum of deflections depending on the strength of the external magnetic field and the magnetic momentum of the atom itself. But quantum physics predicts only two positions; the Stern-Gerlach experiment demonstrates this space quantization.

A beam of silver atoms is passed through a magnetic field and then onto a film plate. The atoms form smudges from which their orientation can be determined. Originally in a mixed state, the silver atoms collapse into one of two orientations once they enter the magnetic field.

Before the computer was used to collect the data, students had to manually move the magnets, read the meter, and record measurements with a painfully slow pen recording device. The process was laborious. Deutsch says that "you could never quite figure out what was going on because it would take fifteen minutes to do any one thing." Now the computer is attached to an apparatus that collects data from the film smear and it does the calculations necessary to determine the shape of the orientations. Two peaks appear, indicating the two magnetic moments predicted by quantum mechanics. Because of the computer's speed, the peaks appear in a dramatic fashion. For Deutsch, this makes the experiment come alive, an example of "something profound and something mundane—that combination, it's like love! The mundane part: something as simple as remembering to degauss the magnet in the experiment. The profound part: the space quantization. That the mundane and profound go together—like washing dishes and love—it's one of those things that is hard to learn."

A third experiment, the Mossbauer experiment, demonstrates resonance fluorescence, a technique used to infer the mean lifetime of the excited state of nuclei. It works in a fairly straightforward way for atomic transitions, but a number of problems arise when you try to use it for nuclear transitions: crystal structures

magnify the movement of individual nuclei and widen the energy curve, masking its natural width; when nuclei emit a photon they recoil and this motion shifts the spectrum. Rudolf Mossbauer discovered that by reducing the temperature one could eliminate the factors that skewed the curve. And it was later found that iron (FE[57]) could give similarly clear results without lowering temperatures.

In the Junior Lab, the nuclei in a sample of iron are raised to an excited state. As photons are emitted and the energy states drop, the spectrum is measured by a photometer and the computer is used to record its data and fit it to a curve whose width relates to the mean lifetime of the excited state. Deutsch believes that the computer makes this experiment, too, come alive, because it is possible for a student to play with the data, take thousands of curves, and develop a *feeling* for the data. Before the computer, nobody did that because it was too much work. "Now, you can ask a question and say, 'Let's try it.'" In other words, the machine does not distance students from data, it brings them closer to it. And for some students, whose style of approaching problems is to play and experiment with them, it makes physics far more accessible. Using computers for real time data acquisition makes it possible for "bricoleurs" to "play" with scientific data in a way that makes science far more congenial for them. The tool that so many of his colleagues feared would alienate students from the real had paradoxical effects. It has the capacity to bring students closer to the data because it allows them to get their hands dirty playing with them.

The instrument Deutsch most often uses in the laboratory is his Swiss Army Knife, a simple, all-purpose tool. He has generalized his ideas about what makes the knife a good tool to what makes a good computer. This translates into a preference for using general purpose computers because of their transparency. They allow students to "look under the hood," understand all the steps, and write their own programs. Deutsch is unhappy about a recent purchase by the Physics Department of some "fancy data analyzers." He would have preferred getting some boards for general purpose computers and letting students do their own programming. Other people want to use the most up-to-date tools. He wants to use the most transparent ones.

Deutsch encourages students to "open the hood" of the computer or the program and write something of their own to personalize the environment. He makes it easy for students to "jump out of the program and go to BASIC, do something of their own and get back in." But his programs, which do allow students to get inside them, also have their problems. They can be awkward, clunky, hard to use. Deutsch accepts that his programs are somewhat unwieldy, but believes that student resistance to them stems from a deeper issue. "Some kids simply resent that

it is not a matter of pushing the button and getting the answer, but they resent it also of the other apparatus. I usually tell them that they probably don't want to be physicists, that they probably want to be engineers or perhaps go into a hyphenated field where one really performs measurements."

For Deutsch the computer is valuable because you can tinker with it, very much in the style of the bricoleur. "When you can play with scale, saying, 'How would it look this way? How would it look that way? Let me enlarge that region,' you can do the same things you have always done, but do them better." Finally, the computer leaves room for its customization. This allows students to make and then work in a thinking environment that fits their own style of learning. This last is no small thing. For Deutsch, it made it possible for him to write. He believes that it can lead to a revolution in other kinds of learning as well.

Part II. The Freshman Seminar: Problems of Error

Like Martin Deutsch, Professor Robert Hulsizer was interested in educational computing for many years before Athena began. When Athena got underway, Hulsizer enthusiastically submitted a proposal to rethink the freshman physics curriculum in the light of what personal computers could offer. He requested equipment, salary support for himself, and funding for TAs and technicians. Athena turned the proposal down—Hulsizer was told that there was a policy against using Athena funds to support faculty salaries. Hulsizer did not feel he could do his project without released time through salary support and he put his proposal aside.

Hulsizer later went back to Athena with a much revised proposal, this time for a Freshman Seminar on the use of computers in physics. He asked only for funding for equipment and TAs. This second proposal was funded. The design of the seminar was simple: it taught students BASIC and then moved on to using computers for problem solving, using numerical rather than analytical methods.

Hulsizer ran the Freshman Seminar for two years. When he retired in 1985, Professors Robert Ledoux and Stephan Meyer were asked to take over the course. Ledoux's formulation of its continuing primary goal: "to let students learn how to use the computer, show them the power of solving problems numerically, the power, you might say, of 'just plotting stuff.'" He feels successful in these goals. The seminar puts students in touch with physics that they would have otherwise only been able to read about.

Ledoux explains why: Only a small subset of real world physics problems are solvable by purely analytical methods. Most require experimentation, where you do trials, evaluate forces, and fit data to a curve. The computer makes such numerical solutions easier to do. And in a practical sense, it makes many of them possible for the first time. For example, in the Freshman Seminar, a student used the computer to take and plot very small time samples in an independent project on radioactive decay. Ledoux admits to being surprised that the student was able to make it all work. "But it did work and the computer was indispensable."

The Ledoux and Meyer seminar established a continuity with what Hulsizer had begun, but put a new emphasis on problems of physical measurement. "To be a good scientist means you must understand how to make a measurement and understand its limitations," says Ledoux. But he fears that MIT students have an insufficient understanding of uncertainty and the margin of error involved in experimental physics.

> Steve and I both are totally committed to the idea of being experimentalists, that the greatest deficiency in undergraduate education is that people don't really understand how the computer is used to collect data and even if they do, they don't really understand what it means to make a physical measurement.

Ledoux and Meyer use the computer to bring students closer to problems of measurement. For example, in one exercise, they set up a computer program that simulates a ball dropping in space. Students could vary the ball's weight and the height from which it was dropped. The program was designed to display the falling ball, record relevant data, and then allow students to reduce and analyze it. In this exercise, Ledoux and Meyer stressed the importance of the error inherent in the data by having students consider the influence of wind drag and inertia on the measurements.

> In the ball dropping exercise, the problem was simply stated. Measure the acceleration due to gravity and then state the uncertainty. And this blows their minds totally. People are used to textbooks. They are used to computers. They don't understand that the computer is marvelous to take data and reduce it, but what does it mean to have an error? If someone said, 'g is 9.81,' that's totally meaningless. 9.81 plus or minus what? 9.81 plus or minus 10 is not a very good measurement. And so we try to drill into them the whole idea of error analysis.

In the Freshman Seminar, the computer is used for its "paradoxical effect." This tool usually associated with precision and rules brings students closer to the

messiness, complexity, and irregularity of the real world. Ledoux, like most of his colleagues, is eloquent in making the point that "simulations are not the real world."

> There is no substitute for knowing what a kilogram feels like or knowing what a centimeter is or a one meter beach ball. But these things have to be taught to students, and we, faculty should teach it to them. I think we can. I think we have a total obligation to do so. They can't go into the real world without facing this, so why leave it to luck? We have greater wisdom. We should teach students how to take computers and use them properly.

As Ledoux sees sit, computers help students see things that would be otherwise inaccessible. But he, too, points out the double movement: the computer's speed and ability to simulate reality can also mask it. The fact that things can go either way, is, for Ledoux, the best reason to make sure that computers are part of the curriculum. If the faculty does not teach students how to use computers correctly and judiciously, "the students are going to use them incorrectly, but they'll use them."

Transparency and the Back of the Envelope

For years, it has been a commonplace among MIT science and engineering faculty to say that students seem to be losing a sense of scale, of order of magnitude. Many associate it with the abandonment of the slide rule which demanded that its users insert the decimal point themselves in favor of calculators which make no such demand. There is no question of going back to slide rules, but Ledoux believes that in the calculator and computer culture, good pedagogy demands that students be forced to do "back of the envelope calculations" where you need to understand the scale you are working in, the units you are using, the number of significant digits that make sense. He feels that students have gotten lazy; they don't want to do things by hand and they don't want to include units in their calculations. The effects are unfortunate. In this realm, a lack of understanding error now can translate into a "space shuttle blow[ing] up" later. He sees the job of the faculty as being a kind of "mind police," making sure that students do understand.

> In any model you have to state error. It's an unpopular but fundamental topic. I think it is something in which MIT education is very deficient .... I've never heard a student say, "I took a course in error analysis." I've never seen that. .... There may be some in applied math, but for physical measurements, you have to learn it through a lab. So we try to provide that service. We're ideologues in this; we're preachers.

Ledoux shares Martin Deutsch's belief in transparent tools. The physicist needs to be in touch with the apparatus, "you have to get in there, mess with the data, really know your equipment." For him, the presence of the computer in the lab dramatizes this point. When something does not work in an apparatus, such as an oscilloscope interfaced with a computer, students begin by assuming that the problem is in the computer. But, he continues, "the problem is rarely there." It is "usually something trivial," something mechanical: a power switch not turned on, too much noise, wires not connected. The physicist needs to know how to sort out all of this.

With the exception of one project, in the Freshman Seminar students do all their own programming, including graphics programming. For Ledoux, the programming is in the service of bringing students closer to problems of estimation, scale, and error. "When students plot points for the first time, they literally understand what the physical screen is, what the graphics screen is, how to actually put points on the screen," says Ledoux. Even the limitations of the computer screen can bring students closer to the real. If a curve drawn on the screen and a theoretical curve don't look the same, one is led to investigate the screen's resolution. "If the two things overlap, they may differ at the tenth of a pixel level, which you can't see." And this tenth of a pixel level may be of critical importance. It may be the margin of error that makes all the difference.

In the second half of the year, Ledoux and Meyer have seminar students work on individual projects. Students design, perform, and analyze their own experiments. They write their own programs to assist them in data collection and analysis. These programs are written in BASIC which Ledoux and Meyer feel is the best language for this purpose. But BASIC was not on the original list of approved Athena languages. After much protest from the faculty, Athena agreed to support BASICA, the IBM version of BASIC. Ledoux and Meyer use "Quick BASIC," Microsoft Version 2 because they believe it is far superior for their educational goals. As Ledoux puts it, "We would have been fools" not to use it. The Physics Department purchased the Quick BASIC for use in the seminar, an example of what many faculty see as doing an "end run" around Athena or even

making their Athena projects work "in spite of" Athena. The Physics faculty criticize Athena for favoring IBM and DEC products. In their view, Athena has put vendors rather than educational considerations first. Ledoux and Meyer pay a high price for their principled decision to use the non-IBM BASIC. They have to keep track of the software, update and maintain it. "It takes manpower to do something that Athena doesn't support."

Part III. Simulating the Invisible Physics

In the 1960s, Judah Schwartz and Edwin Taylor were part of a research group headed by Jerrold Zacharias whose focus was innovation in physics education. Schwartz and Taylor continued this work, collaborating on films which illustrated principles of relativity about which we can have no intuitive sense. One of these films had a particularly strong impact on their colleagues. It demonstrated wave packets propagating as a function of time and fragmenting upon collision. Anthony French describes the reactions of his colleagues to the original film.

> The first time our senior theorists saw that, they were amazed. It shows a wave packet moving along and suddenly reaching a barrier or some potential. It reaches this thing and it fragments and you see this. You see this thing moving along and doing remarkable things. That's something the likes of which had never been seen. It is difficult to calculate that out without a computer. Seeing it as a moving picture is also wonderful. There's been published recently a handbook, a picture book of quantum mechanics which includes, among other things, some of these wave packet collision things, but to see them just static doesn't begin to give the effect of watching a screen and seeing these things actually happen.

Taylor has continued to work on relativity and quantum physics, now using interactive computing. He wants to go beyond the passive presentation possible in a film to give an experience of *living* in the quantum world, including the ability to perform experiments and see how things work within it. In other words, Taylor is trying to create *microworlds* in which the rules of quantum physics and special relativity apply. Taylor's computer programs are designed to demonstrate the invisible physics. Taylor wants students to develop intuitions about it the way they develop intuitions about classical physics: by manipulating its materials.

One of his programs, for example, simulates what it looks like to travel down a road at nearly the speed of light. Shapes are distorted; they twist and writhe.

Objects change color and intensity. All of this can be described through the laws of physics, but Taylor points out that "you can't experience them directly, except through the computer." Because Taylor shares with his colleagues a reticence about how much students can learn through passive demonstration, he is committed to using the computer in an interactive way. The students are required to manipulate the models. The "demonstrations" are never ends in themselves. "As far as the students go, these are utilities, and I assign exercises for which they need them."

Athena funded the development of Taylor's relativity programs which he uses in his course on special relativity. But since Athena sees its mandate as funding projects that will have a major impact on undergraduate instruction, it has made further funding contingent on Taylor's being able to demonstrate that his programs will be used in large scale teaching efforts in Physics. Athena has asked the Physics Department to assist in evaluating the applicability of Taylor's programs to other courses.

The discussion within Physics about Taylor's programs created a forum for strong views about how computers can help and hurt. In general, Physics faculty are opposed to anything that smacks of demonstration rather than experiment. As physicists tell it, demonstrations are the stuff of engineering education. An engineer is satisfied with a simulation. A physicist wants to be in touch with the real. Canned programs, even as sophisticated as Taylor's, seem to them "demonstrations." They fear that such programs will taint the scientific culture with engineering values.

John Negele, for example, fears that Taylor's demonstrations give people a feeling of understanding, but without their having developed real insight. To him, the elegant demonstrations can give a false sense of security. Deutsch, with his aesthetic of the transparent, fears that simulations always function as black boxes. Students watch them as they would a movie. He fears that people substitute models for reality and come to believe that something will happen in the real world because they have seen it in a model.

In fact, Taylor tries to safeguard against the possibility that students will passively "watch" his demonstrations by insisting that students use them as utilities. But like his colleagues, Deutsch tends to use the discussion of Taylor's work as an occasion to object to simulations in general. In Deutsch's case, simulations run counter to his style: "I like physical objects that I can touch, smell, bite into .... The idea of making a simulation ... excuse me, but that's like masturbation." Deutsch characterizes his views as extreme, and characterizes Taylor's work

as good in its genre. Indeed, he laughingly accepts a characterization of it that it is a "good thing of the bad kind."

Although physics faculty stress that the scientist's primary commitment is to experimental data and an intimate knowledge of the instruments that put you in touch with it, they accept that simulation can be a necessary evil since our contact with the real world can only put us in touch with the world of classical physics. John Negele points out that as a child, he learned Newton's laws by playing with baseballs, but there could be no such direct access to the world of quantum physics. Relativity and quantum physics may need an artificial aid to visualization. In other words, when physicists look at Taylor's work, they see it as an exception to a cherished rule.

They are willing to accept a simulation when no "real world" experience could possibly be substituted, but when Taylor uses the computer to demonstrate something that could be done in a traditional laboratory setting, one sees the full force of his colleagues' feelings about keeping the computer in its place.

The following interchange between John Negele and Robert Ledoux, spoke eloquently to this issue:

> *Negele*: One of the problems that a physicist has to come to grips with is that sometimes light behaves like a particle and sometimes it behaves like a wave. If you have a dike and two little openings, the waves of water will propagate through those two little openings. They'll form little rings, which will then interfere with one another, and you'll see the results of these two wave fronts coming out, interfering with one another. That's a very clear wave front. If you think about shooting bullets through these two holes, you know the bullet goes through one or through the other. Now, you're being told as a student of quantum mechanics that sometimes you're supposed to think about light in one way and sometimes you're supposed to think about it the other way. And so a very important experiment comes to mind. You take this case of two slits and decrease the level of illumination so you're very sure photons are only going through one at a time. You would be tempted to say, "Well, by gosh, this is going through one at a time. It's like a bullet, it goes through there or it goes through there." And there's a dramatic demonstration which can be done to show in a way which just hits you over the head in a beautiful way, that even though they're going through one at a time, they are managing to diffuse. It's a fantastic experience for a physicist who is beginning to think about quantum mechanics. And I think many of us have the same reaction, that to simulate that on the computer ....

Ledoux: It's a cheat.
Negele: It's almost sacrilege.

Reticence about simulation is born out of physicists' allegiance to the "real." But it also has another source. The inspiring lectures and mathematical commitments of great MIT physicists have become the stuff of legend. Physicists are trying to protect a particular style of instruction, the elegant and analytical lecture, from a technological intrusion. There is, too, the fear that by making numerical solutions too easy, computers will dull the mathematical edge that is the mark of the great physicist.[2]

Because Taylor wants to develop software that will be of use to other people teaching relativity, he has faced a conflict with Athena that has not come up for his colleagues in the Physics Department. Deutsch wrote software, but he is the first to say that if he was starting over today, he would be buying it off the shelf from a commercial vendor. Negele, Ledoux, and Meyer are all using software developed by someone else. Taylor, in contrast, is trying to create a new product. To be influential, it has to run on widely available machines. He is frustrated by Athena's emphasis on cutting-edge technology.

> MIT's Athena is bound and determined to do things in the most elegant way possible. For one thing, they have the professional graphics adapters. Well, the professional graphics adaptor is a product of IBM that nobody ever bought. It was a very nice thing, but it was too expensive, and so it's sort of like the dinosaur. No one has bought it. All the ATs that MIT provides have professional graphics on them, and none of my colleagues anywhere in the world use them. The thing they want to use next is the RT. The RT is another delightful machine that nobody's using. No physics professor has an RT. If I program on that I am programming myself out of the general use of these programs. Now, they are going to talk about the Athena work station as the be all and end all. It will be extremely powerful and located all over the campus, but nobody else in the world is going to have them. This is my big problem: the ability to produce programs that other people can use and still meet the requirements of Athena that I use their fancy machines.

[2] Of course, allegiance to the analytical lecture as an inviolate form does more than keep computers out of the lecture hall. Anthony French points out that it keeps demonstrations out as well: "I believe there should be a much bigger emphasis on seeing and experiencing real phenomena."

Part IV. Institutional Roadblocks

Professor John Negele begins his comments about Athena with a blunt statement: "I think what I'll finally end up telling you is how it has *not* happened in Physics. He goes on to graphically illustrate why things haven't happened by contrasting his Athena experience with the way Cal Tech handled a student of his who went to teach there. The example takes a special drama from the fact that the Cal Tech approach allowed his student to write a textbook that Negele is using in his Athena teaching at MIT.[3]

> I'd like to tell you how that book [The Cal Tech textbook in computational physics] came about and why you will not find any book like that at MIT under Project Athena. At the same time as the first meetings were occurring here with Project Athena, his chairman at Cal Tech told him that they were interested in a serious effort to get some serious intellectual involvement of physicists using computers in undergraduate education. He didn't want menus which gave students a convenient way of doing repetitious exercises. He wanted something with real, conceptual substance. So they said, "We'll give you a year to work on this instead of your normal teaching obligations. The deal is, you will spend your time working on this. We will give you two full-time TAs. We're not going to ask you to write a proposal to some committee, go there and defend it, waste a lot of time with bureaucracy. We're going to give you a computer in your office; we're not going to say you have to go to some cluster. We are going to let you use whatever language you want; we're not going to say you must use this language or may not use that language. We are going to tell you from the start that anything you write, you own: hardware, software, a book, whatever. We're not going to get into the hassles of deciding who owns it, of fee splitting, etc. We want to do everything humanly possible to make something of intellectual substance happen at Cal Tech. Furthermore, there is going to be a course requirement and every Cal Tech undergraduate is going to have to take one from a small list of courses which satisfy the computational requirement. So when you develop this course, you can be jolly sure that there will be people who are going to take it, because that's the only way that a physicist is going to satisfy the requirement."

[3] Steven E. Koonin, *Computational Physics*, (Menlo Park, CA: Benjamin/Cummings Publishing Company, 1986).

In this narrative, through a study in contrasts, Negele touches on several of the starting conditions that Athena put on MIT experiments in educational computing. At least as the Physics Department came to understand policy, Athena would not release faculty time by paying academic year salaries. Proposals needed to be defended to committees. Faculty members were told that Athena machines could not be in their offices, but had to be in public spaces. And Athena's guidelines for who owns marketable products are unclear: IBM and DEC have rights as does MIT. It is certainly not the case that faculty members own the fruits of their labor in any simple sense.

In addition, Negele's narrative touches on a less tangible condition for participation in Athena: you had to go along with the centralized and bureaucratic "Athena style." In the intellectual climate of the Physics Department, this was more than offputting; it was intolerable. Negele recounts that when Athena was first being discussed, he and a group of colleagues from Physics approached Professor Felix Villars, chairman of the faculty. They asked Villars why there was no real faculty participation in the planning for Athena. Negele reports that the reply to their question described a closed door: Villars "had asked about this and had been told that the negotiations with the vendors, with IBM and DEC, were too sensitive to allow the faculty to get involved."

Negele says he found this answer "personally offensive." He did not feel it was acceptable to sidestep faculty in planning a major educational innovation. Negele involved himself in Athena as early as he could, but even so, he felt that the key decisions had already been made. "They created a huge number of committees and I was a real sap and it took me about a year to figure out what these committees were about. These committees were to give the impression of involving faculty. But there was no substance. Nearly every major decision had already been taken."

In fact, while serving as a member of a technical issues committee chaired by Jim Bruce, an issue came up about which Negele had strong feelings. It concerned the Athena rules for which language and operating system could be used. Athena insisted that it would not fund projects that used BASIC or DOS, while FORTRAN and UNIX were acceptable. When the question of these restrictions came up in the Bruce committee as a "technical issue," Negele exploded. For him, this was not a "technical" matter at all, but a form of censorship. "It was as though they were saying, 'these are the approved books and these other ones have too much secular humanism in them or whatever.' It is anathema. It has no place in the university."

> If my department chairman told me what book I must use to teach a course, I would hand in my resignation. As a professor of physics, I believe I am the person who can make the best judgment as to what textbooks to use when I teach a course. Now who are these people at Project Athena? And who do they think they are that they should tell me what languages I'm allowed to use to teach a course and what languages I'm not allowed to use?

Negele says that the result of his outburst was that he was asked to "cool it," because this issue, for better or worse, was closed.

In fact, Athena would later soften its language restrictions, ironically because of technical difficulties and increasing pressure to get things running. But this relaxation of the rules was several years in coming and many faculty were alienated in the process. In Physics, it is clear that the rules about language and operating system were important in creating an environment openly hostile to Athena. As Negele sees it, they led his colleagues to a basic assumption that "Athena is something they don't want to hear about. They think it's such a stupid, ridiculous, terrible mistake. They aren't even going to waste their time talking about it."

Negele is a faculty member with a deep commitment to educational computing and thus to Athena's basic idea ("to make this new technology available to students.") He wanted to become involved; he did become and stay involved. But he feels that Athena has inhibited good things from happening. In his judgment Athena policy makers have always shown poor educational instincts. And their lack of sufficient faculty consultation has not made it easy to check them.

For example, Negele thinks that in Physics, the required senior thesis is a natural place to put computer resources. And yet, either because it would not lead to the production of software for vendors or because it was not part of the original Athena "model"

> there was a great resistance to using Athena machines for senior theses. They were supposed to be used for large courses. That was crazy. At that time, there was nothing of intellectual substance that had been developed for large courses, but yet there was a whole generation of students coming up with senior theses and UROP projects.

Negele has similar feelings about Athena's position on word processing. He sees it as a commonsense application that saves students time, and for some, seems to increase quality. And yet, Athena found word processing too mundane to support.

As Negele sees it, intellectual substance got lost in Athena's bureaucratic shuffle. We have seen that in his evaluation of the Athena experience, it is very important that no textbooks have been produced. He also feels that the majority of proposals, although "nice," are lacking in intellectual substance. "They don't blaze the way." "They don't put MIT in a position of intellectual leadership in a way that I would like to see."

Negele makes a distinction between applications which use the computer to do more efficiently something that you already know how to do and applications where the computer makes possible something new. In his opinion, it is the second that "turns students on." But at Athena, "empire building urges were so strong that everything else got lost and we got a whole bunch of clusters and no intellectual substance. It is really sad."

Negele believes that he has lost the chance to do significant work in computational physics: the text already exists. In his personal research he finds it more fun "to move into a new field where I'm the first person to do something than to move into a mature field where the cream has already been skimmed off by other people." He generalizes his sense of missed opportunity to MIT as a whole; even if Athena policy could be reversed, it is too late. Because to the degree that MIT shares his intellectual personality, the "timing is no longer right" since "other places have seized the lead."

Despite his harsh criticism, Negele's attitude toward Athena is constructive. He believes that the straightforward acknowledgment of a missed opportunity is a first step toward doing things better. But not surprisingly, his suggestions for how to do things better are radical in relation to where Athena began. In brief, it is to "abolish all those silly rules." Indeed, one might look at the "Negele Plan" as a suggestion that MIT switch to the "Cal Tech Plan," that is to say, identify faculty members who are going to do something, put a machine in their office, give them one to take home, give them a semester or a year off to work, and tell them that they own their product. In other words, give faculty incentives that will make it worth their while to be taken away from their major research. Negele believes that decisions about what to fund and who should evaluate it should be decentralized and put into the hands of department chairmen. "The chairman knows his faculty and he knows he can trust them .... In Physics, for example, you are dealing with world class physicists, experts in their fields. They just don't need oversight."

Negele points out how important it was that Cal Tech required courses that used educational computing. He contrasts its situation with that at MIT. MIT

students "optimize time" by taking required courses and a few graduate courses. With rare exceptions, elective courses are doomed to low enrollments. When enrollments in Negele's computational physics course dropped, he had a "sign up" sheet for sophomores. But although thirty-five people signed a petition asking him to offer the course, only ten of them were able to fit this elective into their schedule because the Physics Department had just added a new requirement for its major. Negele thinks that courses which integrate educational computing should be Institute requirements. Students could choose among what was available—the architecture students would tend to go in one way, the physics students in another. But as students "made one or more natural couplings," faculty who had put in the time and effort would know that there was a clientele for their work.

For Negele, the abolition of restrictive "rules," a system of faculty incentives, and the creation of an environment where students do not pay a price to participate are good first steps. But Athena must go further than these: "They've got to stop spending money on equipment and start spending the capital on intellectual production." Negele knows that Athena is considering a possible future as a reliable and stable technical utility, a good "facility." Negele clearly thinks that this modest path has an advantage over trying to be an arbiter of intellectual taste. But "given the choice between ideas and facilities, I always go for the former." Becoming a facility carries with it the danger that too much emphasis will continue to be placed on the technology to solve educational problems.

> These people have labored under the illusion that all you've got to do is to get this wonderful software environment and have a graphics capability and a word processing capability, and then all of the problems will be solved. If we could just get a little more software, a little more hardware, the next generation of computers. Actually, a very primitive machine is adequate for almost all you ever do. An off-the-shelf PC or an off-the-shelf Macintosh is all it takes to run the software in the computational physics book. I don't think MIT should be in the business of creating this empire of trying to write the software that vendors haven't managed to write. That's secondary. What MIT ought to be doing is leading the field intellectually. The companies are going to beat us in the software. We ought to put the undergraduate's tuition money into something the private sector can't and won't do and that's the intellectual issues.

Negele takes an Athena proposal in the Astronomy Department as an example of what it means to capture serious intellectual issues. To study astronomical

events, astronomers have traditionally taken pictures of them. Today, charge couple devices provide astronomical images and store digital information about them. The Athena project that has won his admiration digitally recorded a recent supernova and put it on a disk. "Students can now analyze it instead of canned data in some lab. They can superimpose the image of the night before with the night after digitally subtracted, and see what it looked like ... with all of the different color separations you could use and so on." This project allows students to profit from "the fact that a wonderful astronomical event has happened—that he or she can play with data that hasn't yet been published. That's got to turn people on, that's got to show them what it's really like to do modern science."

Part V. The Student Perspective

Parallels with Faculty Views

As a science, physics is a kind of religion. Physicists believe in it; physics students believe as well. Interviews with students are striking in the extent to which they mirror faculty attitudes. AT MIT when students declare physics as a major, they enter a culture that has a well-worked-out and highly coherent discourse for talking about itself. There is, for example, a shared way of talking about what distinguished physics from engineering. And on this subject, student interviews echoed faculty perspectives. For example, in the words of one junior, "What I like best about physics is the sense of dealing with the universe. It elevates you to an eternal sense. In engineering you're dealing with man-made objects. The goal is to create new devices of human significance. In physics you're relating to absolute truth as opposed to practical truth."

Students also share their faculty's ambivalent stance towards computers. Like their teachers, they are wary about introducing engineering technology into the purity of their scientific enterprise. On the other hand, students, like faculty, appreciated that the computer opens up a new physics. The ambivalence was clear as they talked about the laboratory experiments and how they have changed with the use of the computer for data collection and analysis.

Before the computer, access to the data in the Mossbauer experiment on resonance fluorescence was through Polaroid photos taken of the oscilloscope.

> Before you could only get a qualitative understanding. There was less data on the photo than you have on the computer. Now, you can get an exact fit of the data to the function and see the deviations and how much of it doesn't fit an exact curve. Seeing that the data fits in spite of the variations is part of the allure of physics.

But students are wary of losing touch with the real data by letting the computer do things for them. One says, "If you don't know what you are doing, it's easier to be mindless on the computer. You put worthless data in and get worthless results out, but think it's noble somehow just because it has been through the machine." "Taking the data by hand has its advantages although you can only get about one fourth of the data points that way. Doing it by hand forces you to think about the data as it is being taken. You have to know what you are looking for." For this student, working through a long series of calculations by hand allows him to keep track of the kinds of errors he makes. His dedication to the aesthetic of transparency is complete: "When you work on the computer it is hard to tell what data is 'lose-able' and what is essential. The computer makes these choices for you. That's why I do it by hand as much as I can. Doing it by hand, point by point, you can see subtle variations in the phenomenon you are studying."

One graduate student TA sums up the trade-off in much the same terms:

> When you chart tables by hand you get a sense of where you can push the data. That's how you get an intuitive sense for an experiment. The feeling is built up in a numerical way because you live with these numbers so much that you begin to understand what they mean. It's nice to be able to enjoy your graph and take your rule to it. With the computer, it's a mental process further removed from reality. The problem is in getting the feeling of moving the ruler on the paper with the computer.

The Experience of Junior Lab

Students in Junior Lab appreciate Martin Deutsch: they recognize his love of physics, commitment to teaching, and the intellectual passion behind his investment in the Junior Lab software project. And students appreciate how Deutsch's software protects the immediacy of their contact with data. "The software was designed so you can still see the raw data. We're still doing our own data analysis. It's a tool that doesn't change the data fundamentally, just does the things

you'd know how to do anyway." The TA in Junior Lab makes it clear that the program is written so that students are required to think.

> People have to calibrate the computer as they would an oscilloscope. They have to understand the error intrinsic in converting from analog to digital. They do not just punch a button and get a result. Because the program does not have bells and whistles, it compels students to figure out what has happened. For an engineer, when an instrument doesn't work, he'd send it back to the factory.

But, the TA stresses, echoing what we have heard from Deutsch, physicists can't work like that. They need an intimate knowledge of their instruments to stay close to their data.

Despite a general appreciation for Deutsch, particularly by those students who share his stylistic preference for transparency, some feel that the "homebrew" nature of his software is getting in their way. The complaints are numerous: it has bugs, a hard-to-work interface, there is not enough space on the disks to run the programs and store the data. Students try to get around these problems by transferring files from the PCs to the larger Athena system when they do their analysis. But this is technically difficult, impossible for all but the most persistent and talented students. Several students made it clear that the combination of poor documentation and temperamental programs translates into sleepless nights and a lot of lost time.

Students sometimes interpret the primitivity and the lack of documentation of the software in Junior Lab as an expression of the faculty's fear that the computer will mask a direct perception of the phenomenon under study. One student puts it this way: "They won't give you information about the lab because they think it'll 'diminish' the experience. But [what they are giving you] is only one cut above blank paper."

Student comments made it clear that although they have their benefits, Deutsch's experimental systems are taking their toll: "Sometimes the program would load the data and sometimes it would not. I don't know what the problem is, but it seemed like whenever Professor Deutsch came into the lab, he'd hit some buttons and it would work. When you did it on your own it wouldn't work."

The idea that a program's author is party to a mysterious knowledge that makes things work is common when people deal with "hackers," virtuoso programmers who like Professor Deutsch have a joy in the computer that is a "love of the machine for itself." There is a cost to seeing a program's author in this

way: a feeling that at times only the "wizard" can make it all work. Deutsch has put enormous effort into giving students the sense that they can be masters of their data, their instruments, their analysis. But the challenge is finally to build a system that is as transparent to its users as to its creator.

The Freshman Seminar

Students in the Freshman Seminar are excited that so early in their MIT careers they are getting a feel for "cutting edge" physics. They are aware that the computer allows them an access to problem solving that would not be possible without it. One student talks about his sense of discovery of numerical solutions, another reports that the computer allowed him to understand gravitation for the first time. It helped him break things down into small and manageable parts.

The seminar teaches about problems of measurement and error by asking students to account for the effects of forces that are not factored into pure theory, for example, friction and air drag. The method seems to work. One student says: "When you have to consider these forces that otherwise you ignore in dealing with plain theory, it changes things radically. You really try to make the connection between theory and data. It's not someone else's experience. You have to make it work for yourself."

Student comments about the Freshman Seminar, like student comments about Junior Lab, mirror the faculty's concern that computers should not function as black boxes. Much like their professors, physics students use the idea of the black box to distinguish themselves, as scientists-to-be, from their image of the engineers-to-be. For one, "Using computers as a black box isn't right. For scientists who are interested in understanding phenomena and theorizing, it's important to know what the program is doing. You can't just use it to measure everything." For another, computer mastery is an important element in identity: "Of course you can't know everything about the computer [as a computer science major might], but you should know enough that you don't have to hand your work over to a Course 6 guy."

And like the students in Junior Lab, there is some ambivalence about what is gained and what is lost with the machine. Some insist that the computer allows students to set the pattern in data, the patterns that allow theory to come alive. For example, when the computer allows something like time and voltage to be instantaneously demonstrated in relation to one another, there can be a better understanding of the curve it generates. "You're not just seeing the curve drawn, you're seeing it actually happen ..., created point by point." "If you do the same

things 1,000 times by hand, you lose the sense of what you're doing. It takes so long you forget the goal. You lose the forest in the trees." But on the other hand, as one freshman reflects, "drudgery" can keep you close to the data in a tactile and tangible way.

> There is some truth to the importance of getting a feel for the data by hand. When you plot it with the computer, you just see it go 'ZUK' and there it is. You have to look at it and think about it for a long time before you know what it all means. Whereas when you draw it, you live with it and you think about it as you go along.

Student interviews make it clear that two people using the same computer system are not necessarily two people having the same computer *experience.* People respond differently to a system that demands transparent understanding. People differ in their response to the computer's speed. Some can use it to see things they could not see before and others feel left behind and need to compensate by using pencil and paper to "catch up." Traditionally, software evaluation has focused on "ease of use." The study of Athena in Physics makes it clear that this is much too simple a parameter. More research is needed to understand styles of learning in order to effectively mobilize the personalization of learning that computers make possible.

Part VI. Conclusions: Powerful Ideas or Paradoxical Effects

Judah Schwartz is involved in an Athena-funded software development project in the Concourse Program and is chairman of a subcommittee of the Project Athena Study Group which is looking at the impact of Athena on the residence system. He began a conversation about Athena by recalling a lecture he had attended: Barbara Tuchman on "How I Write My Books." Schwartz recalls: "The first sentence out of her mouth was 'First you have to have an idea.'" He continues:

> You need to have an idea, a powerful idea, in order to do something that is conceptually interesting. Physicists have been using computers forever and will continue to use computers forever. It doesn't make a conceptual difference in what they do and so they are very casual about computer use, and I think properly so. The question of how to make a conceptual leap is much subtler, and with far fewer answers. It's something

> whose major intellectual impetus derives from trying to understand something about how people learn about abstraction and the building of quantitative models. MIT doesn't have such an enterprise.

There is an important sense in which Judah Schwartz is right about the current state of educational computing in the Physics Department. Where the new efforts with personal computers have departed from the old "number crunching models" (what Schwartz is referring to when he talks about physicists having "used computers forever"), they have proceeded without 'powerful ideas.' They are not based on new concepts of how people build quantitative models or deal with abstraction. They have, in general, used the computer for data collection and analysis. But our study of how computers are being used in Physics challenges Schwartz's idea that in order for something interesting to happen you have to have started with a powerful idea. Perhaps counterintuitively, there seem to be powerful effects from relatively mundane implementations of computer technology.

First, by relieving the tedium of data collection and data plotting and data analysis, phenomenon are revealed in a sharper form. Students are brought closer to theory. The fact that they can make quick calculations allows them to implement thought experiments which give more of a feeling for what is going on than they had ever had before.

Second, the computer makes it possible to solve classes of problems that are only accessible through numerical methods. This means that there is a new challenge to the traditional "textbook physics," a physics skewed towards those problems that can be solved through analytical methods. As this learning experience with numerically solvable problems becomes more widespread, there is a possibility that there might be a change in what is considered "high physics," prestigious physics.

Third, in a seeming paradox, the computer brings students closer to the "real" by forcing a new consideration of error and the limitations of measurement. With the computer, students are better equipped to handle uneven and anomalous data. Before the computer entered the laboratory, if a student's one round of an experiment yielded only anomalous data, the student could not bring his or her experimental result in relation to theory, but would have to rely on prepackaged data. Physics faculty stress that the "real world" is not neat and the scientist's primary allegiance is to messy data. This was always a problem for students who were meeting theory for the first time and needed "clean data" in order to see the theory it demonstrated. In sum, the computer, by being a magnificent tool, becomes more than just a tool.

Fourth, the computer forces a sharpened and more nuanced discussion of simulation and whether and how it differs from "demonstration." What is the status of simulation when it is interactive? What is the educational potential of being able to manipulate interactive microworlds on the screen? The question of what kinds of mental models people are building on the basis of their experiences in such microworlds is yet to be studied.

Fifth, the computer allows for its personal appropriation. As people make it their own in their own way, different intellectual approaches are revealed. By giving students an opportunity to sculpt a thinking environment that "fits," computers may facilitate learning. For example, those who need to "play" with data in order to learn a subject matter have enriched possibilities as do those who function best in a transparent learning environment.

The Athena experience in Physics makes it clear that to push educational computing further, junior faculty must be given greater incentives to participate. For example, Martin Deutsch was free to experiment with educational computing because he comes to it as an older, successful senior faculty member, in his words, "without ambition."

> I don't have to write two papers a year, three papers a year. I don't have to write any papers a year if I don't feel like it. It's easy for me ... in a funny way, and that goes into many things. It's easier when you're sort of pulled out a little bit, out of the struggle.

As things stand now, major commitments were made by senior faculty who can "afford to play" (such as Deutsch and Hulsizer), and by younger faculty (such as Ledoux and Meyer for the Freshman Seminar) who were recruited. But these younger faculty are going to "cycle out" of these assigned roles. In part, this is because educational computing is not the center of their research concerns or a major passion of theirs (as it is for someone like Deutsch) and in part because of an incentive system which will ultimately "punish them" if they stay in too long. Junior faculty know this. One says, "If you want to have innovative teaching initiatives, which you should, senior faculty should take it on themselves to initiate it. You can draw on the junior faculty, but you can't ask junior faculty to initiate large programs, because it's death to their careers."

The Athena experience in the Physics Department has some clear lessons for thinking about the future of educational computing. In microcosm, it illustrates the ways in which Athena has inadvertently constructed roadblocks to the kind of innovation it might well have encouraged. A few examples make the point. Physics faculty were reticent to experiment with the lecture format. They wanted

it to be inviolate. But is is also fair to take them at their word when they say that they had no exciting ideas about how to use computers there. In this context, when Athena gave the impression of "pushing" the use of computers for large lectures as being more educationally valid or exciting than other uses, it led to resentment. Professor George Clark captured the sentiment of the Physics Department on this: "There was a sense and I think there still is a sense that there was an attempt to try to force the use of computers in situations that didn't seem appropriate, like using computers in lecture situations. There was willingness to support applications that would somehow introduce computers into lecture demonstrations and so on. Some of that seems like nonsense."

Professor Alan Lazarus, chairman of the Physics Department's education committee, described how the physicists were met with an uncomprehending Athena establishment. The physicists wanted to

> get the kids involved as quickly as possible. We wanted to use BASIC not because we thought BASIC was better than anything else, it was just what the kids knew. And Athena was very, very adamant about that being a bad thing, we should not do that. We finally got them to agree that it was okay but then they disowned us. They said, 'Here's the terminals but don't expect us to keep them up to date, or put new operating systems in them. You're on your own. This put people off fairly severely.

Athena's ban on BASIC inhibited several of the Athena projects. Even today, with loosened language restrictions, we have seen that faculty who use BASIC have feel they are accomplishing something important in spite of Athena.

In Physics, it is fair to say that Athena's technical decisions about language and hardware have inhibited rather than encouraged faculty involvement. Edwin Taylor struggles to stay off the "prestigious" Athena machines because he doesn't want to develop software that his colleagues will be unable to use because it is made to run on machines they will never have. And in fact, many exciting results from Athena experimentation in Physics have come from relatively mundane computer use that could have used off-the-shelf materials. Athena has created a climate in which some people experience this statement of fact as a sign of failure.

In Physics, the centralized, bureaucratic Athena structure went profoundly against the grain. The Physics faculty consider themselves and their discipline in a position of intellectual leadership at MIT. They are used to being consulted on important matters of educational policy. They are sensitive to the differences between their field and engineering disciplines and were not happy to be given pedagogical advice or directives from outside of their discipline, and particularly

not from the School of Engineering. Finally, the experience in Physics dramatizes how from the perspective of a group of faculty with a long history of interest in education, the Athena organization could seem uninterested in questions of educational impact. As Professor Lazarus puts it, speaking of the development of the Freshman Seminar,

> There was no feedback. Athena seemed to care less about what we did with the terminals or what we had developed. Nobody came around and said, 'What have you guys done, show us, give us a demonstration.' We have invited people over, but nobody, as far as I know, came. We mentioned this to the people at Athena and they said, 'We have no provision for that sort of thing.' It was clearly 'Here they are, good-bye.' It was not the kind of mutual working together it should have been.

There is no reason to think that Athena treated Physics any differently than other departments. But in Physics, Athena was dealing with a group of faculty who are deeply involved in a tradition of innovation in education. The stage was set for a confrontation and disappointment. The Physics Department has profited from the Athena presence. There was an impetus to try some things that might otherwise have been put on the back burner. The department has used Athena to learn a great deal about educational computing. Athena can learn from the reticence as well as the enthusiasms that the Physics Department took from that experience.