The CATESOL Journal

BRIAN LYNCH
University of Melbourne
PETER COUGHLAN
University of California, Los Angeles

Making Use of Computer-Assisted Language Learning in Higher Education: A Report from UCLA

An earlier version of this paper was delivered as a Featured Speech at CATESOL '93, Monterey, CA.

■ This paper presents an overview and analysis of a three-year computer assisted instruction (CAI) project conducted at UCLA. The project, funded by UCLA's Office of Instructional Development, had as its primary goal the development of materials for individualized instruction within the ESL service courses. In the paper, we present a brief description of how the project was carried out (including an account of the development of one piece of software), and a discussion of some of the major issues which arose concerning our implementation of computer-assisted language learning (CALL) or computer assisted instruction (CAI) in a university ESL setting. We will be presenting this discussion not as CALL experts, but as ordinary ESL teachers and administrators exploring a new technology. Rather than a state-of-the-art report on CALL, then, we intend this to be a portrayal of one experience that will hopefully be of use to those who are considering the implementation of CALL in their own instructional settings. Our discussion will refer to several sources of qualitative data that were collected over the three-year life of the project. These were written documentation produced by the project teaching assistant (including memos and journal entries), other written documents produced across the life of the project, and interviews and questionnaire data collected from the ESL service course teachers after the official completion of the project.

Introduction

The following paper is an overview and analysis of a 3-year computer-assisted instruction (CAI) project conducted at UCLA. The title of our paper refers to CALL—computer-assisted language learning—while our project was referred to as CAI (computer-assisted instruction). One way of distinguishing between the two is to use CALL to refer to student-accessed, lab based work, where the computer and student interact with no (or minimal) instructor intervention; and to use CAI to refer to the use of computers in a classroom environment to assist instruction by the teacher. We recognize that this distinction can blur, however, when you consider the computer lab as a classroom, or when the result of computer-assisted teaching is seen as language learning. In this paper, we will use the term CALL to refer to all the types of language teaching and learning with computers that we attempted in the context of our project.

We wish to make it clear at the outset that this paper is not a discussion by CALL experts, but by ordinary teachers and administrators trying out a new technology. As such, it is not meant to be a state-of-the-art report on CALL[1]. We hope, however, that it will provide a meaningful portrayal of one experience, and that the successes and failures we encountered may be useful to administrators and teachers hoping to implement CALL in their particular instructional settings.

It may be helpful to briefly state why we embarked upon a computer-assisted instruction and language learning project. One motivation was our desire to respond to a call for proposals from the Humanities Computing Committee. This committee was charged with—among other things—encouraging the use of computers in undergraduate education at UCLA. We also felt that our ESL program, in order to keep pace with developments in second language education, needed to make some effort at investigating the potential of CALL.

Project Overview

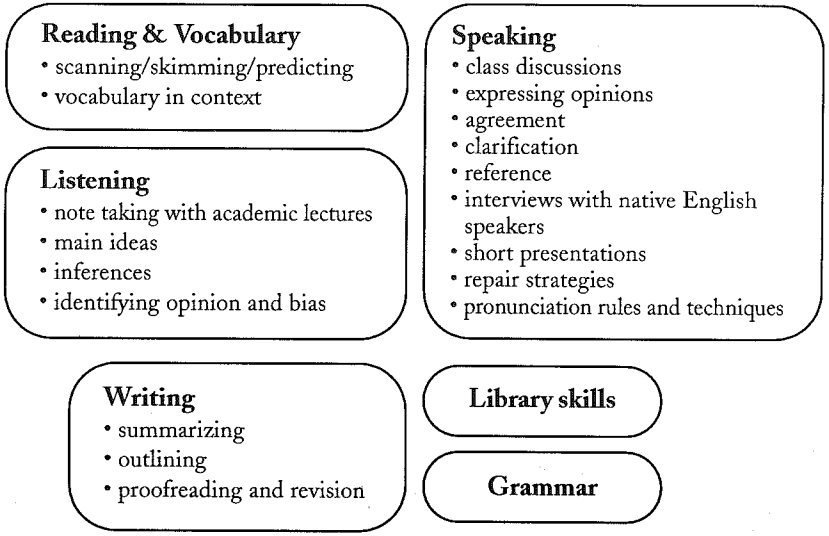
Establishing preliminary objectives

Undertaking this project led us to ask a fundamental question: what could we do with computers that we weren't already doing in the ESL service courses? One area we identified was the need for more individualized instruction, since our teachers had been complaining for years about not having the time to cover the curriculum and attend to individual needs that varied across the student population. Sending students to the traditional (audio) language lab hadn't worked—they (and their instructors) didn't seem to be motivated in that environment. One-on-one conferencing (especially in our composition classes) helped, but class size and teacher responsibilities limited the use of this time-consuming strategy. We felt that it would be helpful if we could send our students to a place where they could get individualized, interactive help with their language problems and that perhaps CALL could provide such an environment.

This, in turn, led us to think about more specific aspects of CALL that would be important to take advantage of—what would be the most innovative and motivating directions to go in? We decided that we needed to explore applications that went beyond printed text and to work with sound and images as well—in other words, the range of possibilities currently referred to as *multimedia.* Our ideal goal for CALL was to create an individualized, interactive environment in which students could self-access language learning situations relevant to their needs and interests. It was our intention that any materials developed through the project would serve as a complement to our normal classroom teaching, rather than an alternative or replacement.

When our proposal received funding, implementation of our preliminary ideas began. We hired a project teaching assistant (TA) (at 50% time), a technical consultant who did HyperCard programming and graphic design (at 25% time), and several part-time research assistants who helped in running the computer lab, developing software, and so forth. We began

Figure 1.
Originally targeted learning and teaching tasks



the project with a couple of open meetings in which faculty, teaching assistants (TAs), and other interested students enrolled in our TESL/applied linguistics program brainstormed ideas for aspects of our curriculum that might provide the basis for software development. Various learning tasks and teaching objectives (presented in Figure 1) were identified from these early discussions.

These tasks were envisioned primarily in terms of software that would be used by the students in our instructional computing lab (the Humanities Computing Lab, a Mac-based lab-classroom with 20 networked individual stations and a teacher's station with projection capabilities). We were also interested in using the computer as an instructional tool inside a normal classroom, through projection of the computer screen via a liquid crystal display (LCD) and overhead projector (OVH). We identified the following ways in which the computer could be used in the ordinary classroom setting:

- to project samples of student writing
- to record student responses on disk for later work/display
- to play back task-related listening segments
- to display pictures and text for introducing new topics

While the LCD/OHP combination provided a relatively easy and rapid way to introduce computer technology into the ESL classroom, the implementation of these classroom activities was initially hampered by the unavailability of the necessary technology from UCLA's campus-wide media center. When an LCD finally became available for our use, access was limited by awkward check-out procedures (including a two-week advance notice requirement for classroom use) and competition from departments across campus. In the final year of the project, we purchased our own projection equipment, allowing our ESL instructors much greater and more flexible access to the technology.

Project Accomplishments

In the first year of the project we concentrated on developing software in HyperCard[2] that would capture some of the tasks and objectives we had targeted in our first meetings. By the end of the first year we had produced prototypes for the following stacks: (a) a *timed/paced reading* stack, which allowed students to measure their reading rate of a provided text, and answer multiple-choice questions after completion of the reading; (b) *grammar exercise* stacks to study and practice article usage and count/non-count nouns, and a *cloze builder* which would allow teachers to create stacks designed to practice various grammatical features; (c) a *vocabulary* stack, which was intended as both a tutorial and reference to provide students with contextualized vocabulary from other exercises; (d) a set of *Pronunciation Bingos,* game stacks which allowed students to practice auditory discrimination of minimal pairs; and (e) a *Word Stress Game*, which allowed students to practice stress placement in multisyllabic words.

During the second year of the project, we worked on refining and debugging stacks developed in the first year, based on feedback we got through testing from ESL teachers and students. We also completed class sets of the Pronunciation Bingo stacks and timed and paced reading stacks, and develop prototypes for the following stacks: (a) two *prewriting exercises,* which helped to elicit and refine ideas for writing; (b) an *academic listening* stack, which was designed to help students comprehend conversation between an academic counselor and a student; and (c), a *home* stack, which provided an easy-to-use index to the software that had so far been developed.

Figure 2.
Overview of accomplishments

	Year 1	Year 2	Year 3
Develop	*Prototypes* of • grammar exercises • Pronunciation Bingo • timed/paced readings • vocabulary tutorial/ reference • word stress game	*Prototypes* of • prewriting (cubing and looping) • academic listening stack • home stack	
Refine expand		• timed/paced readings • Pronunciation Bingo	• cubing and looping • home stack • timed/paced readings
Facilitate access		*For students:* • ESL "open lab" hours • word-processing class *For teachers:* • training sessions	*For students:* • Expanded ESL "open lab" hours *For teachers:* • stack development class • lab and classroom support • documentation

As can be surmised from the above descriptions, we spent a good deal of our time and resources developing software during the first two years of the project. Over the course of the second year, however, it became clearer to us that we needed to spend more time on training teachers and students to use the hardware and software, as well as to document and otherwise make the software more accessible. Thus, during the final year of the project, our main goal was to attempt to make the use of CALL in the ESL Service Courses a resource that did not rely on the project staff (who would no longer be funded) for its realization. Toward that end, we produced and refined documentation on the use of computers and the project software both in the Lab and in the classroom, worked more intensively with teachers who wanted to try CALL out, but needed some training or guidance to do so, refined our computer-based index to the software, and held workshops on basic Mac skills, the ESL stacks, and HyperCard programming for interested teachers and students. Figure 2 summarizes the project's accomplishments over the three years of its existence.

Development of a CALL Stack

Because software development used such a large percentage of our available resources, we will outline the development of a stack in order to give readers an idea of the process of creating a new piece of software. Our decision to select specific stacks from the many first year ideas for development beyond the prototype stage was based on two criteria. First, we were looking for stacks which could be used by teachers of different levels within our ESL curriculum. This meant looking for classroom activities common to multiple levels, so that the greatest number of students could benefit from our software development. Second, we wanted to develop stacks which could be used by individual students outside of regular class time in order to free up class time for other activities. We wanted this software to improve upon those classroom-based activities (the idea being that the computer-assisted environment could provide something that the teacher or the traditional classroom setting could not).

The timed and paced readings, which fit these two criteria, seemed a perfect choice for further development, and it is these stacks which will provide an example of what for us was a long and arduous process of developing a piece of software suitable for classroom (or computer lab) use.

Background on timed/paced reading stacks

At UCLA, timed and paced reading exercises are currently used in two levels: ESL 33B (high intermediate ESL), and 33C (advanced ESL), both of which meet for approximately four and one-half hours per week.

Student enrollment in sections of 33B and 33C accounts for roughly 40% of total enrollment in ESL classes offered by the UCLA Service Courses. The timed and paced reading exercises are based on reading materials taken from Fry (1975) and Spargo (1989).

In the classroom, the timed and paced reading activity proceeds as follows: students are first paced through a text at a speed determined by the teacher (generally about 200 wpm at the beginning of a quarter, gradually increasing throughout the quarter). During the exercise, students' reading speed is controlled by the teacher, who calls out the remaining time at 15-second intervals. Students then answer comprehension questions about the text. A second, longer text is then read, this time self-paced by the students, who are instructed to read for between 70 and 80% comprehension, as measured by their performance on questions which follow the reading. Answers are then checked in an answer key and performance charted on a graph, a practice which allows students to see their progress over the course of the 10-week quarter.

While the timed and paced reading exercises were generally well-received by ESL students, two problems with the exercises as they were practiced in the classroom were readily apparent. First, the exercises were quite time consuming, requiring at least 20 minutes each week. TAs often complained that this was a large amount of time to devote to a single exercise, when they had so much more to cover in the curriculum.

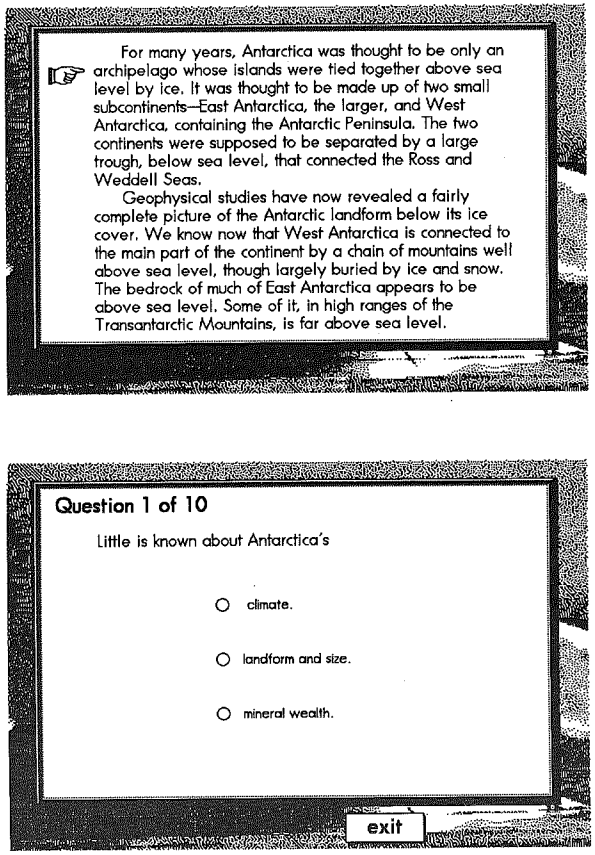
Second, due to the teacher-led nature of the exercise, students were forced to read at a classroom-averaged speed, though the classroom reality was that some students were already reading at 400 wpm, while others were more comfortable at 150 wpm. This meant that students at both ends of the reading-speed continuum had to compromise their reading speeds in order to maintain a uniform class rate. We believed that a set of computer-based timed and paced reading stacks could address both of these problems: first, by taking the exercise out of the classroom (thereby freeing up class time); and second, by allowing students to individualize their reading rates at the computer (thereby allowing each student to read at her/his most comfortable rate).

From Prototype to Class Sets

The prototype on which the final version of the paced readings are based was created by one of our program's TAs with a fondness for computers. The prototype stack is simple—it divides a text up into screen-size chunks, and displays these chunks for a certain length of time before moving to the next chunk of text. When the text is completed, multiple-choice questions appear on the screen, with immediate feedback provided to responses.

After initial testing, a critical design problem was discovered: with no way to prepare readers for a change in cards, users were unable to pace their reading. In subsequent versions of the stack, we therefore explored ways to facilitate this pacing. First, we tried a scrolling text, which turned out to be too difficult to read — a fixed text is necessary. We then reverted to a fixed text, accompanied by an audible beep which sounded at regular (five-second) intervals. This proved to be too distracting, since readers concentrated more on counting beeps than on reading text. We next tried a moving pointer, accompanied by a beep. This feature proved the best solution yet,

Figure 3.
Current version of paced reading stack



though some teachers complained that the noise was unnecessarily distracting. In the current version of the paced reading stacks, therefore, a silent moving pointer moves line by line down each screen of text. (See Figure 3).

While seemingly easy to detect in retrospect, such design flaws are quite invisible during construction of a stack. What is of primary importance is getting the stack to work correctly (with the programming knowledge available to you); only after you have a working stack can you turn your attention to design problems. As more people use the software, these problems become easier to spot, but it is not easy to get people to use your software if it is still in rough shape. That is, people (especially students) are less willing to spend time with software that is still in development.

For this reason, you must, at a certain point in a software development project, make the miraculous leap from a single working model (which can be tested out by a willing individual) to a classroom set which is both substantial and stable enough for a teacher to consider worth trying with a class. In our case, that meant designing, developing, and piloting a companion stack to the paced reading we had built (since the exercise depended on both types of readings), and developing class sets of timed and paced readings to last an entire quarter. In other words, we needed to develop a total of 40 reading stacks in all, if we were to convince teachers that what we had to offer could take the place of the paper version of the classroom exercise.

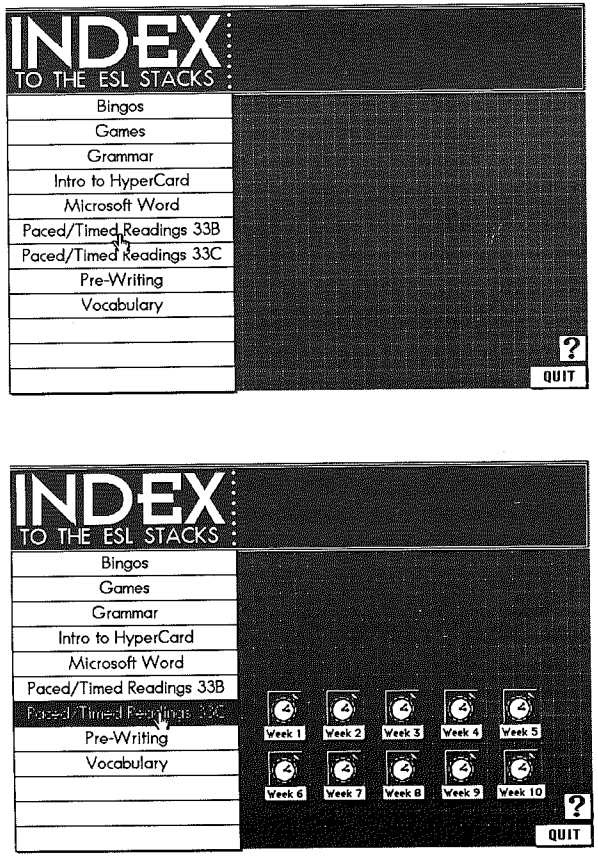
Once you have managed to build up a critical mass of exercises, you are then in the position to invite teachers to use the software and discover how it works under lab conditions. Though we had the software running flawlessly on individual computers, we encountered numerous problems when using it in a lab full of students. These problems had to do with technical problems which arise when running HyperCard off a server on a network with 20 computers. We had not originally taken this performance aspect into account since our original conception of the software was that it would be used on a drop-in basis, by one student at a time. However, because the first teacher who offered to pilot the software did so on the condition that she bring her entire class to the lab to do timed-and-paced readings, we were obliged to try and meet her demands.

In addition to the technical problems we encountered in running the software in the networked environment, we also noted a number of further design problems in the stacks. For instance, the subtle audio feedback provided to a single user produces quite a cacophony when multiplied by 20—not exactly ideal reading conditions. Furthermore, such public feedback on correct or incorrect responses (no matter how quiet) was embarrassing to students, whose colleagues could hear that they were making errors (even

though students who read at the proper rate were supposed to get two or three errors per exercise).

In addition, we discovered that many of our users were complete novices with respect to the Macintosh. Students (and some teachers) did not know basic routines such as how to insert a floppy disk, open a stack to begin an exercise, or even find a stack on a disk. They had little control of the mouse and inadvertently clicked in the wrong places, opened hidden script fields, typed over text, and otherwise made it clear to us that a stack developer's idea of an intuitive interface is much different from that of a neophyte computer user.

Figure 4.
ESL software index: Access to paced and timed readings



Based on hours of observation, as well as student and teacher feedback, we modified the stacks yet again, this time concentrating on making them "people proof," that is, ensuring that no matter where (or how many times) they clicked, the program would not get damaged or produce an error message. We also created an index which would enable students to find the week's exercise in the simplest way. By clicking on two buttons—one button to select the timed and paced readings for their level, and a second button to identify the proper week's exercises— the students' assignments would open up for them on the screen. This index appears in Figure 4.

In the second year of the program, we had two TAs doing timed and paced readings with their classes. In addition, one TA sent students in on their own time during ESL "open hours", a block of time in which our students could use the lab free of charge in the presence of a project person familiar with the software.

In spite of our successful development of 40 bug-free exercises, the timed and paced reading stacks—and, for that matter, most all the other software developed during the project—has received little or no use since the project officially ended. The picture for overall CALL use is not entirely negative, since a number of TAs have held class sessions in the computer lab, sent students to work with CALL software, and used the LCD for computer projection in the classroom. However, we were puzzled as to why the software we developed would receive so little use from both teachers and students.

The next portion of this paper attempts to solve this puzzle, by addressing a series of issues which appear to have been of importance to our project, and, by extension, to the success or failure of CALL in other higher education settings such as ours.

Issues

Hardware Limitations

One of the issues that we confronted early on in our experience was the limitations that were placed on our original goals by the level of hardware support that was available at UCLA. While it may be at first surprising that an institution with the reputation of UCLA would have a lack of such material support, two factors should be kept in mind. The first is what we hope is a short term effect: the state budgetary crisis. The second factor has a longer tradition at most universities: the tendency of the humanities to lag behind other segments of the university in receiving computing and other technical support. The specific ways in which our project was hampered by these limitations are as follows.

A major disappointment for our project was the failure to develop software that made use of interactive sound and video media. During the first year of the project, we had to work within the limits of the hardware that was most easily accessible to our staff—Mac SEs and one Mac cx. We attempted to find other hardware on campus that would allow more sophisticated multimedia explorations but without much luck. Even our ability to use our Mac SE in the classroom with an LCD for overhead projection was hampered by the fact that there was only one such device available through Audio Visual Services at UCLA. By the time we discovered its existence (a few weeks into the first quarter of the project), it had already been checked out by other departments for large blocks of time. In addition, we needed to install a special video card in our Mac in order to have it work with the LCD. As a result, not much was done with this option during the first year. During the second year we continued to encounter hardware limitations that prevented us from exploring multimedia. The one interactive media station at our institution was either unavailable or not working most of the year.

Another instance in which hardware limitation affected our ability to carry out our ideas involved our academic *listening stack.* The objective of this stack was to make use of an existing series of audio tape recordings of a UCLA student counselor talking with an ESL student. We wanted to create a sound stack that would have a series of layers, incorporating different types of activities to be used with the counselor-ESL student interview recording. After building a preliminary version of this stack, we very quickly ran into the limitations of storing sound on a hard drive and began to investigate the possibility of pressing a compact disk to store the audio data. Ultimately, we gave up on this stack idea because the hardware requirements—even with sound stored on a CD—outstripped our machinery. The sound in our last version of the prototype needed to be broken up into five second chunks, and even then it did not run smoothly, especially in the computing lab where we wanted our students to be able to eventually use it.

Until the end of the project, use of the project software in the Humanities Computing Lab was greatly hampered by the frustrating amount of time it took for the workstation computers (Mac SEs) to access the software from the server. This was due to the fact that the lab was set up with machines that were not powerful enough to compensate for the slowness of the network design. At the beginning of a class, it took 15 to 20 minutes to have all the machines up and running in either HyperCard or Microsoft Word. Because of this, the instructor would need to arrive at the lab 20 minutes before class time and remain with one program for the entire class (since changing from one program to another would require another 15 minutes of down time). Several teachers mentioned this as a reason that kept them from bringing their classes to the lab more often.

In the summer following the final year of the project, thanks in part to some heavy lobbying on our part with the Humanities Computing Committee, the lab was upgraded with Mac LC IIs, which improved the speed of software delivery to the workstations somewhat. However, as with the previous lab hardware, these machines are among the least powerful computers that Apple is currently producing.

Student and Instructor Attitudes Toward CALL

As previously mentioned, the need for training of both teachers and students became a priority during Year 2 of the project, in our efforts to make greater use of CALL. Our sense was that some of the strongest attitudes mitigating against the use of CALL reflected a lack of familiarity and comfort with the technology. We attempted to investigate this more systematically with a survey of student computer literacy and by tracking teacher use.

If we assume that *hours per week* of computer use is a reasonable indicator of level of computer literacy, then the results of a survey which we administered to students indicate that slightly more than half of undergraduate ESL students are at least moderately computer literate (i.e., they report more than one hour of computer use per week). Results for graduate ESL students show them to have a somewhat higher level of computer literacy than undergraduates, with approximately 75% reporting regular use.

In terms of teacher computer literacy, it seemed that very few of our instructors had much experience with computers during the first year of the project. As a result, the project did seem to introduce more teachers to computing in general. We organized introductory computer workshops in an effort to make more of them feel comfortable enough to try out the lab and classroom possibilities and made an introduction to CALL materials a part of the TA orientation program. During Year 1 only two or three teachers made occasional use of the lab or accepted our offer to have the project TA do a demonstration lesson in the classroom. In Year 2, increased efforts at training and providing support for teachers resulted in a noticeably greater number of TAs using CALL: four (out of a total of 16) TAs made use of the lab on a weekly basis during at least one quarter (either sending their students to do the timed and paced readings or by holding class in the lab to work primarily on composition), and several others used the lab for one- or two-hour sessions, either to provide their students with an introduction to computers, or to explore the various ESL stacks.

Another aspect of our training efforts, directed at students, involved hiring a research assistant from the department to act as a monitor/resource person for ESL students during the weekly ESL open lab periods we had arranged with Humanities Computing (four hours per week). To make the use of the lab outside of arranged class hours more affordable during the first two years of the project, we also arranged for discounted lab use cards for ESL students, which we subsidized out of our grant. However, the reports we received from our teachers suggested that, except for those who used the lab for their word processing needs, students felt that even the discounted, $10 per quarter lab fee was too much to pay, especially on top of the textbooks and other materials that they were required to purchase for their ESL classes.

During Year 3 we refined the documentation begun earlier that described for teachers how to make use of the lab and ideas for using the computer in the classroom via the LCD with overhead projection. TA use increased slightly over the year, with five TAs trying CALL for the first time and four others continuing their use from the previous year. At the end of Year 3 we were able to purchase our own LCD device with OHP, upgrade the existing TA Mac, and mount them on a cart as a portable workstation. This year's orientation program for TAs included a demonstration of this workstation, along with an introduction to the ESL software.

Still, even with our increased efforts at training and support, there were indications that more was needed. For instance, one of the teachers who had most consistently used the lab and ESL software said in his end-of-project interview that he had been "working in the dark" in the lab. The project TAs journal, on the other hand, notes that this same teacher was "relaxed about the technology" and that the teacher "seem[ed] to feel that there [was] a way out of every problem."

Time

Related to the issue of computer literacy and the need for training is the problem of time. The general sense that making use of CALL tended to be an extra burden on their time was a reaction common to many teachers (in fact, of the nine teachers interviewed at the end of the project, seven of them commented on the fact that using CALL tended to take up more of their time than traditional lessons, particularly because of planning in advance and the slowness of the network server in the lab). One of these teachers said:

> I don't plan enough in advance to take advantage of the CAI software . . .that's another level of planning on top of just finding the written text itself that I want the students to be manipulating.

This same teacher felt that with the availability of the LCD for overhead projection she would be interested in using the computer for instruction more in the future.

The Nature of CALL Tasks and Software

One obvious aspect of the nature of CALL tasks and software is that they are different from pencil and paper tasks. Our students are reading off of a CRT rather than the printed page. One issue that arises is the difference between reading a computer screen and reading the printed page. Our experience leads us to believe that, in terms of reading to edit text, some of our teachers felt that reading off the computer screen is significantly more difficult than editing on the printed page.

Perhaps with writing activities, given the predominance of the computer as the vehicle for academic written work, the use of CALL has more validity than other tasks, such as those involving grammar practice. Our teachers found CALL useful for activities such as working with students on paraphrasing—-projecting the original text and then eliciting paraphrases which were then typed onto the screen—-and paragraph development and revision. What is common to the nature of all of these CALL tasks is the ability to project student and teacher work from individual workstations to an overhead visible to the entire class, the ability to save this work on disk for future use, and the ability to individualize the task (as with setting the reading rate in the timed/paced reading).

Related to the nature of CALL tasks are the issues of software-led versus teacher-led activities and interaction with the computer versus interaction with other students. The timed/paced reading stack again provides a good example. As one teacher said in a recent interview:

> I was put off by the individual nature; [with CALL] you lost the class spirit aspect as you do it together, plus you miss what I think is a healthy competition that develops at the tables when doing it in the classroom—I have seen this really help individuals... when one on one at their computers, it's a whole different atmosphere.

This teacher believes that, at least with the timed/paced reading activity, too much is lost by having it presented by the computer rather than the teacher. The teacher has the ability to cheer the students on and to give them on-the-spot consciousness-raising advice concerning their reading strategies. However, this teacher also commented that her reaction was not limited to the reading software alone:

> My philosophy is that I think it's more important to have student-student interactions, rather than students interacting with the computer.

Other, though not all, teachers commented on the notion that a dynamic element of their traditional classrooms—the student-student interaction—seemed to be lost in the computer lab. This may have been due primarily to the original design of the lab — rows of workstations that kept students apart. However, this configuration was redesigned by Humanities Computing after the final year of the project, thanks to the ideas put forward by our project TA. Now, the workstations are clustered, and can lend themselves to students interacting with each other across their workstations. At least one teacher felt that using computers in the normal classroom with the LCD helped to promote student interaction and was positively received by them.

Effectiveness of CALL

An issue that has been central to discussions of CALL is its effectiveness. A key question here is: what criteria do we use for judging effectiveness? As with program evaluation, in general, the literature on CALL effectiveness has focused on student outcomes (See Dunkel, 1991). Do achievement scores improve significantly as a result of CALL? The only data from our experience that speaks to this criterion is inconclusive: in the case where one teacher consistently sent her students to the Lab for the Timed/Paced readings, without teacher assistance, the supervisor for that level noted that the students did not appear to achieve the same dramatic reading rate gains witnessed in the other classes in which the teacher conducted the reading activity in the classroom. These student gains were not quantified or compared in any systematic fashion, however.

Another criterion for judging effectiveness, perhaps more important than trying to analyze outcomes, is student and teacher attitude, one of the issues we have already discussed. Comments from several TAs including one who used the prewriting software—-"it works just as easy without computers"-—indicate the link between teacher attitude and the criterion of *efficiency.* As seen in the previous discussion of time, teacher attitudes are likely to be negative to the extent that CALL is seen as extra work, a burden on their already overburdened schedules. Student attitude, on the other hand, may be different. Our data indicate that students, where given the opportunity, have responded quite positively to CALL.

In the analysis of outcomes as a criterion for judging effectiveness, CALL is generally compared to traditional, paper-and-pencil based instruction. We have briefly mentioned certain differences between CALL tasks and paper-based tasks. This suggests that comparison, in the sense of the traditional experimental research design, may not be the most appropriate way to evaluate CALL effectiveness. Perhaps the most useful and convincing evidence will come from naturalistic studies of the use of CALL across a variety of settings. What we have attempted to do here is the beginning of such an analysis. We need to go back to the data we have already collected and to continue to talk with the teachers who have tried to use CALL and continue to use CALL in our ESL program. We hope to have more to report on this experience in the future.

Conclusions

We would like to conclude with a set of questions to consider before embarking upon a CALL project such as the one we attempted. These questions come from the experience we have tried to outline in this paper, and we hope they will help others to have the most productive and positive experience possible with CALL in their setting. Some questions to consider before embarking on a CALL project:

1. How computer literate is your faculty? If they are not computer literate, what interest do they have in becoming so? (Only instructors who are comfortable with computers will use them and encourage their students to use them.)
2. How computer literate are your students? What are their perceived computing needs? Do these needs overlap with aspects of their ESL instruction?
3. Is there adequate technical expertise at your institution (hardware, software, printers, maintenance) in order to accomplish your CALL goals? Is there adequate access to computers for both students and faculty?
4. Can you clearly articulate a need for CALL that goes beyond what your existing curriculum is already doing? Are there CALL activities that can improve upon existing practice to the extent that justifies the time and resources needed to develop and make use of them?

These are questions that are not always asked by those who encourage and fund CALL projects and are not always considered in enough detail by those who attempt such projects. We believe that these questions will lead to the kind of discussion necessary to honestly assess whether you have a real need for CALL and whether the support and commitment is there to sustain its use. ■

Brian K. Lynch is a lecturer in applied linguistics at the University of Melbourne. He was formerly academic director of ESL at UCLA. His primary research insterests are in program evaluation and language testing.

Peter Coughlan is a Ph.D. candidate in applied linguistics at UCLA. His dissertation research examines the language that children use during problem solving at the computer. Some of his other interests include the relationship between language and cognition, and the application of activity theory to linguistic research.

References

Dunkel, P. (1991). The effectiveness research on computer-assisted instruction and computer-assisted language learning. In Dunkel, P. (Ed.), *Computer-assisted language learning and testing: Research issues and practice (pp. 5-36). New York: Newbury House.*

Fry, E. (1975). *Reading drills for speed and comprehension. Providence, RI: Jamestown.*

Pennington, M. (Ed.) (1989). *Teaching languages with computers: The state of the art. San Diego: Athelstan.*

Spargo, E. (1989). *Timed readings* (3rd. edition). Providence, RI: Jamestown.

1 For a thorough overview of the field of CALL, see Dunkel (1991) or Pennington (1989).

2 HyperCard is an authoring system for the Macintosh environment. HyperCard stacks (or individual applications developed with HyperCard) are likened to stacks of index cards. Cards which contain different pieces of information are linked together by different buttons. Clicking on certain buttons brings a user to a new card (containing related information).