An Integrated Framework for CALL Courseware Evaluation

Philip Hubbard
Stanford University

ABSTRACT: The evaluation of courseware for CALL is one of the more challenging tasks a language teacher is faced with. Currently, most evaluation schemes consist of either a checklist or a list of questions to be answered. This paper offers an alternative approach to evaluation in the form of a flexible framework from which teachers can develop their own evaluation procedures. The components of the three major sections of the framework—operational description, teacher fit, and learner fit—and their interactions are described and some suggestions for use are offered. The evaluation framework described is one module of a more comprehensive framework incorporating courseware development and implementation as well.

KEYWORDS: courseware, software, evaluation, framework, approach, methodology

Introduction

Courseware evaluation has been around as long as courseware, but with the spread of microcomputers and relatively inexpensive commercial programs in the early 1980s, it became necessary for large numbers of classroom teachers to learn how to evaluate courseware themselves. Following the lead of textbook evaluations, forms and procedures of various types were developed, often consisting of nothing more than checklists. While these could help a teacher to select appropriate courseware and recognize its strengths and weaknesses, they could also be misleading. For example, the checklist originally promoted by the CALICO Journal, for all its detail, failed to mention the word "language" in any of its substantive questions and tended to promote certain types of learning at the expense of others (Hubbard 1987).

In retrospect, it is not surprising that the early efforts in courseware evaluation would have been somewhat less than ideal. The medium was new and its qualities, especially as they related to language learning, not clearly understood. Some more recent articles dealing with evaluation have shown
considerably more sophistication in dealing with these special qualities of CALL as opposed to CAI in general (Odell 1986, Curtin and Shinall 1987), but they are still basically in the form of a questionnaire, with all its inherent biases and limitations. This article will propose an entirely different approach to evaluation, based on a framework incorporating the relevant components of both the courseware and the language teaching/learning process. After these components and their interactions are described, the use of the framework to evaluate courseware will be discussed.

The framework approach to courseware evaluation is different from others and requires some introduction. A framework in this context means an integrated description of the components of something—in this case CALL materials—with respect to a particular goal—in this case evaluation. Rather than asking a specific set of questions, a framework provides the tool through which an evaluator can create his or her own questions or develop some other evaluation scheme.

This particular framework is a module of a comprehensive methodological framework for CALL outlined in Hubbard (1988a and b), which also includes modules for courseware development and implementation. All three of the modules are guided by four basic principles, which help to determine their form and content and integrate them into a coherent system. Below, these four principles are presented with respect to the evaluation module: similar formulations exist for the courseware development and implementation modules.

- Principle 1: the evaluation framework should be linked to a general framework for language teaching methodology. That is, it should build as much as possible on already established views of how to analyze the effectiveness of the language learning process.

- Principle 2: the evaluation framework should be designed to accommodate as wide a range as possible of methods, teachers, learners, and syllabus goals. It should be flexible and non-judgmental in its basic categories and not biased toward any particular view of the nature of language and learning.

- Principle 3: the evaluation framework should be linked in form and terminology to frameworks for courseware development and implementation. In other words, the evaluation process should not be isolated from the process of developing and using courseware, but should instead comprise just one module of a comprehensive methodological
Principle 4: the evaluation framework should express the multiple dependencies among the various components of CALL. The framework should not be designed to support an evaluation process that operates exclusively in a linear manner because the components clearly interact in non-linear ways.

This paper attempts to realize these principles in a specific evaluation framework: other frameworks, of course, could be developed to be consistent with these principles.

**Overview of the Framework**

In this section and following ones, the proposed framework will be presented, along with a brief description of each of its components and how they interact. The final section will comment on using the framework in the evaluation process.

Principle 1 states that the evaluation framework should be linked to a general framework for language teaching methodology. The evaluation framework to be described below is built on a view of the language teaching/learning process proposed by Richards and Rodgers (1986). Their framework for the analysis of language teaching methods describes any method in terms of three organizational levels: approach, design, and procedure. Basically, approach represents the method’s assumptions about the nature of language and language learning; design represents the realization of those assumptions in terms of curricular goals, learning tasks, and the roles of learner, teacher, and materials; and procedure represents the implementation of those goals through specific learning and practice activities.

Because the Richards and Rodgers framework is designed to analyze whole methods rather than individual pieces of courseware, some significant liberties have been taken to accommodate the rather different goals of courseware evaluation: this will be readily apparent to anyone familiar with their framework. However, their three-way distinction remains an important one, and the modifications to it will in most cases be relatively superficial ones.

Figure 1 lays out the major sections of the framework. At the top is the courseware itself, consisting of the basic software and any accompanying text, documentation, etc. Since the courseware in some sense embodies teacher, technique, and materials, this would fall most readily under Richards and
Rodgers' notion of *procedure*. To clarify this sense of *procedure* within the context of evaluation, it will be referred to as the *operational description*. Below and to the left is the part of the framework involving the roles and goals of the learners and the syllabus which most closely fits Richards and Rodgers' concept of *design*: it will be referred to as the *learner fit*. To the right of it is the section which includes the considerations corresponding to Richards and Rodgers' notion of language *teaching approach*: it will be referred to as *teacher fit*. The section on the bottom left, *appropriateness judgments*, represents the results of the evaluation in terms of the basic fit of the courseware with the teacher/evaluator and learners. The section on the bottom right, *implementation schemes* represents the ways in which the courseware might be exploited consistent with the learner and teacher fits,
including ways not anticipated by the designer. It is another manifestation of Richard and Rodgers’ notion of procedure, and it serves as the interface to the implementation module of the more comprehensive framework outlined in Hubbard (1988a and b). One of the more striking aspects of this type of framework is the presentation of relationships among many of the components in the form of a network. As Principle 4 above stated, the existence of multiple dependencies for certain components of the evaluation process is assumed, and it is necessary to find a way of accommodating them. A network is one way of presenting those dependencies in a single, visual “map” making it easier to see how the components are interrelated. As a general rule for interpreting the network, whatever happens at the head of an arrow depends on what has happened at the tail.

This will be more obvious when the components of the framework are depicted in detail in Figures 3, 4, and 5.

**Operational Description**

The function of the operational description in the framework is to provide an objective and detailed view of the operation of the software and various accompanying elements. The information gleaned at this level is then used in determining the learner and teacher fit.

**Figure 2: Operational Description**

<table>
<thead>
<tr>
<th>OPERATIONAL DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
</tr>
<tr>
<td>Activity Type</td>
</tr>
<tr>
<td>Presentational Scheme(s)</td>
</tr>
<tr>
<td>Screen Layout</td>
</tr>
<tr>
<td>Materials</td>
</tr>
</tbody>
</table>
There are both central and peripheral parts in the operational description in Figure 2. The peripheral parts are seen as external to the basic software and may or may not be present in a given courseware package (with the exception of documentation, which hopefully would always be present). They include the following:

1) accompanying text - the textbook, handbook, or workbook that the software integrates with;
2) documentation - any information outside of the instructions actually incorporated in the software that discusses the program’s design, contents or operation;
3) tutorial - a separate program designed to teach the user how to operate the main program effectively;
4) record keeping - any record keeping utility outside of the main program;
5) other utilities - any other programs for the instructor or user, including programs for authoring materials and/or customizing the operation of the main program.

The software itself is defined here as the program and any materials and instructions whose appearance is directed by it, regardless of whether or not they exist in files on the disk that are separate from the program file. In fact, from the standpoint of evaluation, it is useful to view “program” here as the instructions to the computer for a single learning activity, regardless of whether those instructions are technically part of a larger file, an independent file, or multiple files. The framework in Figure 2 assumes the simple case where the software is driven by a single program manifesting a single activity type, a situation becoming rarer as courseware becomes more sophisticated. For example, the Super Skills disk of Lougheed et al.’s (1987) Reading Strategy Series (reviewed in Hubbard 1988c) contains programs representing three different activities and a number of distinct presentational schemes. In the evaluation process, each of these could be considered independently, and assuming some yielded a better teacher learner fit than others, the students could be directed to focus on those activities and perhaps avoid others. All this means is that to evaluate a given piece of software effectively, it will be necessary to cycle through the framework once for each clearly distinct program.

The first area of the program to be described is the activity type: this term is taken from the framework of Phillips (1985), which has had a significant influence on the present one. Phillips uses the term to classify the broad categories of learning and practice activities which a learner can engage in on a computer. He identifies six activity types: game, quiz, text reconstruction, text
construction, problem solving, and exploratory. To this list could be added other commonly used categories such as drill and practice, tutorial, and simulation, which, though they overlap with the previous six in some cases, may still be useful classifiers.

Under the general descriptor activity type is the more specific presentational scheme. This is an open-ended category used to describe the way an activity is presented to the learners. A reading activity might present timed text plus multiple choice comprehension questions. A grammar activity might present ungrammatical sentences to be edited until they match the original correct form. A vocabulary activity could involve presenting the picture of an object and prompting the student to type in the word within a given time. These are just a few examples of the variety of presentational schemes possible. In some cases, a given activity might have multiple presentational schemes: the preceding example of a reading activity might blend fill-in-the-blank questions with the multiple choice or vary the way the text is presented, e.g. one word at a time, one line at a time, or one paragraph at a time.

The relationship of activity type to associated presentational schemes can be important because presentational schemes that are quite distinct in appearance may in fact belong to the same activity type. Cloze, storyboard (filling in an entire passage of blanks), hangman, word scramble, strip story (where scrambled sentences in a text are unscrambled), and error correction activities (as in the preceding paragraph) are all representative of text reconstruction. In the interest of variety, it may be advisable not to overload a curriculum with one particular type of activity even if the presentational schemes are different.

The presentational schemes by themselves are described in fairly broad terms, but their implementation in a given program depends on a number of other factors, the six most prominent of which are listed next in Figure 2.

Screen layout is the component concerned with how the material presented to the student appears on the screen. This includes such aspects of presentation as print size, spacing, balance, use of color, presence and quality of graphics, relative position of graphics and text, presence and quality of animation, and potentially a number of other considerations. In interactive video, this would also include the quality of the video and the location of text or other video windows. In the present framework, the integration of audio (both speech and non-speech) with the visual image on the screen is also accommodated under this category, though audio could optionally be considered separately.
Timing is a category which may or may not be important in a given program. The most common forms of timing are to set rates, set time limits, or monitor time on a task. For instance, the rate at which information appears on the screen may be set (as with the SPEED command in BASIC), or the rate at which stimuli appear may be set to increase or decrease depending on a learner’s performance. Time limits can be set on the amount of time an image is allowed to appear on the screen or the amount of time a learner is allowed to have to input a response. The time for both of these actions can also be monitored for scoring or other feedback purposes, as can the amount of time a learner spends overall on a given activity.

The notion of control is a complex one that has received some attention in CALL research. The opposing philosophies are aptly described in Higgins (1983), where he distinguishes programs that are magisters, or drillmasters outside of learner control, from those that are pedagogues, or subservient and under the control of the learner. There are in fact at least three potential controllers for any piece of software or any specific aspect of it: the program, the teacher, and the learner. In principle, any of the components in the operational description may be subject to the control of either the program or the user and may be controllable by the teacher as well, but there are several that deserve particular mention because of the different ways they have been handled in existing software.

In a menu driven program, the learner may have the option of choosing a particular activity and a particular presentational scheme. In programs where the material is not strictly sequenced, the learner may have control over what material to select, especially in the case of reading practice and text reconstruction programs. In the latter case, learners may even have the option (usually under teacher control) of producing their own materials. Within a given activity, there are several typical control areas that require exploring in an evaluation. One of these is whether the learner can break out of a particular activity and/or move around freely within it or whether a specific sequence must be followed. Another area concerns whether commentary feedback (e.g. an explanation of why a wrong answer was wrong, see below) is given automatically or only at the request of the learner. The same situation occurs with respect to hints, to be discussed below under help options. These are just a few of the control options that may need to be described in an evaluation.

The next component is input judging. Judging the input necessarily depends on the type of input: letters, words, numbers, sentences, keyboard commands (Return, Space Bar, etc.), a touch on the screen, a movement of a
mouse, and so on. It also depends on what the input is a response to, that is, the format the program is presenting its stimulus in. For instance, if the program presents a multiple choice question, then input judging might include whether both upper and lower case letters would be accepted and whether the distractors would be flagged for possible feedback. If the expected input is a word, then input judging could include wildcard or spelling subroutines, whether anticipated incorrect answers would be flagged and whether alternative correct answers would be accepted. The judging of phrases or sentences as input may involve a parser which could have a number of characteristics to evaluate, including speed of response. Whatever the situation in a given program, an understanding of the nature of its input judging is an important part of the evaluation process.

Feedback is information the program communicates in response to specific input, and it can be of several types. The most common type is simply an indication of a correct or incorrect response, which may be done with words, graphics, sound, or some combination of them. As noted in the previous paragraph, it is also possible to offer information concerning errors of various types that have been flagged by the input judging procedure. Such messages can even be offered for correct answers. Another form of feedback is scoring, both on individual exercises and across exercises cumulatively. Scoring or other records, such as response patterns, can be used as a basis for another type of feedback which directs a learner to a specific exercise for review or perhaps offers suggestions for modifying a response strategy.

The preceding examples deal with explicit feedback, but this portion of the framework may appropriately handle a different type of computer response that could be called implicit feedback. Suppose in a multiple choice exercise that the learner inputs an incorrect answer. Rather than responding directly with "incorrect" the program may simply eliminate that distractor and present the question again. Higgins and Higgins' Sequitur (1987) is a program which offers this type of feedback. Similarly, in a simulation, if the program responds in a way other than the learner expected it to, that response provides implicit feedback about what not to do when such a situation arises again. From an evaluator's viewpoint, the description of a program's feedback should include the full range of responses the computer may provide to a learner's input. The final component of the presentational scheme is the help options. There are two common types of help options: review of instructions and hints. A review of instructions option normally allows a learner to exit an exercise temporarily,
view all or part of the instructions on screen, and then return to the exercise in exactly the same place. Hints, also called clues in some programs, will be broadly defined here as information that helps a learner complete some aspect of the activity successfully. This may include very specific information on a question (e.g. the number of letters in the answer, a list of possible answers, etc.) or information independent of a specific question, such as an on-line dictionary. In the case of hints for specific questions, it is possible for a program to provide multiple types and/or to nest them so that they are accessed in a specific order (often from least to most helpful).

As noted earlier, materials and on-screen instructions are treated independently of the program in this framework regardless of whether or not they are physically a part of the program file on the disk. As noted in Hubbard (1988d), materials are a crucial part of the software whose importance has tended to be obscured by technical considerations in both software development and evaluation. The materials in a program, particularly one lacking authoring potential, require a thorough review if the evaluation process is to lead to a truly informed decision. Instructions, too, should be carefully scrutinized, but there is a crucial difference here. Incomplete or ambiguous instructions can be repaired by a teacher simply by training the students in class in the operation of the program or by producing and distributing hard copies of more complete and precise instructions.

Throughout this discussion of operational description, an attempt has been made to keep the framework at a more or less objective level. It is only when the program operation is viewed through the lenses of learner fit and teacher fit that judgments of appropriateness can be made. This is particularly true of an area like control options, where, as in a real classroom, the amount of control a student is given may depend on the student, the teacher, and the specific activity rather than on any absolute, preconceived notion of who should be in control.

**Teacher Fit**

As noted in Hubbard (1987), language teaching approach as defined in Richards and Rodgers (1986) is an area surprisingly underplayed or overlooked altogether in many otherwise fairly thorough evaluation checklists. As was mentioned earlier, in this framework their notion of approach will be considered in terms of teacher fit, or how well the developer's assumptions about language and learning as manifested in the courseware correspond to those of the evaluator.
Utilizing the framework in Figure 3, the evaluator begins by identifying his or her beliefs about the nature of language structure and function, ideally, though not necessarily, based on some contemporary linguistic theory or theories. Among other considerations this would include assumptions about the roles of rules, patterns, and analogy in grammar and lexicon as well as the general relationship of linguistic form and function. Obviously, it is unreasonable to expect definitive answers to many of the issues involved in clarifying these assumptions, but it is important for language teachers to at least have educated opinions on the nature of language and to have those opinions activated during the evaluation process.

Along with linguistic assumptions, an evaluator needs to be in touch with his or her assumptions about learning in general and language learning in
particular. Richards and Rodgers distinguish two types of language learning assumptions. The first type involves the second language learning processes internal to the learner: the potential impact of age, motivation, first language interference, and so on, as well as the operation of linguistic memory and other aspects of language processing. The second type involves the role of the external environment in the learning process: teachers, classrooms, texts, classmates, media, etc. In both cases a familiarity with current second language acquisition research—viewed through the lens of the evaluator’s experiences as both a teacher and a learner—will yield the most reliable generalizations.

The linguistic and learning assumptions combine in the box labeled language teaching approach. Note that at this point, an evaluator’s views of language learning and teaching are still independent of any considerations of the computer and should be the same ones that underlie his or her classroom teaching. Language teaching approaches as defined here are necessarily idiosyncratic, but in Hubbard (1987) it is suggested that as a step toward describing one’s own approach, it may be useful to consider three common classes of approach—behaviorist, explicit learning/cognitive code, and acquisition/communicative—along with the characteristics typically associated with them. The three differ, for example, in the treatment of learner errors. In a behaviorist approach, error tolerance is viewed as reinforcement of a bad habit, and as such an error is appropriately responded to by noting its incorrectness, modeling the correct form and requiring the learner to say or write it in accordance with the model before continuing. In an explicit learning approach, the error is often explained to the learner in terms of a violated rule or exception to a rule in either the native or target language, depending on the situation and other assumptions held by the teacher. In an acquisition approach, the error may be seen as a part of natural development and ignored if it does not interfere with the content of the communication.

In order to operationalize the language teaching approach in a way that links it directly to courseware evaluation, it is necessary to consider the qualities of the computer as delivery system for language input. The computer is well-known to be particularly good for certain activities, text manipulation, timing, and record keeping for instance, and—at present at least—practically useless for others, such as holding up one end of a conversation. An understanding of these abilities and limitations together with the generalizations of language teaching approach can be used by the evaluator to generate a set of approach-based evaluation criteria. Sets of such evaluation criteria for the three classes of
approaches described above are presented in Hubbard (1987), along with a suggested rating system. A partial list of the evaluation criteria given there for explicit learning approaches appears below:

1) Gives meaningful rather than mechanical practice, contextualized in a coherent discourse larger than a single sentence;
2) Provides hints of various types to lead students to correct answers;
3) Accepts appropriate alternative correct answers within a given context;
4) Offers the option of explanations for why correct answers are correct;
5) Anticipates incorrect answers and offers explanations for why they are incorrect.

While any given approach will have certain areas it focuses on as being important, there are particular issues, such as the treatment of errors discussed above, that all approaches are likely to address. Other such issues include sequencing, the role of explanations, the treatment of learner differences, the nature of learner-learner interaction, the nature of learner-teacher interaction, the nature of linguistic competence, the role of culture, the nature of memory, the role of learner feelings and motivation, and the role of structured vs. unstructured practice. The assumptions in these areas among others form the basis of the evaluator’s approach-based evaluation criteria.

Learner Fit

The third and last major section of the framework is the area of learner fit, which roughly corresponds to Richards and Rodgers’ design. While it is separate from approach in this evaluation framework, it is not entirely independent of it: the design decisions made in setting up the actual class, particularly the syllabus, presumably follow from the teacher’s approach and should therefore be consistent with those assumptions.

A teacher normally pursues the task of evaluation with a particular target group of users in mind. In the ideal case this would be a specific group of students in a specific class, all of whose relevant individual learning characteristics are known to the evaluator. Obviously, such an ideal will not occur in real life, but the more the evaluator can specify about the target audience, the better.
As was the case with teacher fit, the learner fit represented in Figure 4 is a network, with the arrows converging on two boxes: learner variables and syllabus. They will be described first, since together they determine the nature of the learning environment to a large degree. As Figure 4 implies, ultimately, the impact of the other components, such as content and language difficulty, will be calculated with respect to them.

**Figure 4: Learner Fit**

It is worth noting at the outset that all the components of this section other than learner variables, syllabus, and content are taken from the framework of Phillips (1985). That article describes some of these components in greater detail than will be possible here and provides a number of examples of how particular programs can be classified according to certain combinations of categories internal to these components. As with Richards and Rodgers’ work, a few liberties have been taken in adapting it here.

There are a number of learner variables that could be relevant for evaluating courseware relative to the target learners as a group, including, but not necessarily limited to, the following:
1) Age
2) Native language
3) Proficiency level
4) Sex
5) Learner needs
6) Learner interests

In addition, there are individual cognitive style or preferred learning strategy differences which may be considered, such as field dependent vs. independent learners or integratively vs. instrumentally motivated learners, especially to the degree these may correlate with the group variables above: age, native language, and learner interests in particular (see Brown 1987 for a discussion of these and other learner variables). If an evaluator is looking toward truly individualized CALL as a possibility, these considerations become even more important.

Turning to the area of syllabus, which is used here in the wide sense to mean both the goals and objectives and the planned methods of reaching them, there are several different types that have typically been referred to in language teaching literature: structural, situational, notional-functional, and content centered. Another central area of the syllabus is the skill or skills being focused on: in dealing with particular skills, such as reading or writing, the syllabus may be oriented toward the learning and practicing of sub-skills and strategies for more effective processing and retention or production. In such a case it may be structured quite differently from the more general types listed above. In addition to evaluating such typological compatibility, however, there are a couple of other considerations of importance. The first of these is where CALL fits into the syllabus for a given course or program: is it to be an integral part, a required independent part, or an option? Given the limited amount of available courseware, if it is to be an integral part then the courseware may end up determining the syllabus to a large degree. A second and related question is what the role of CALL will be. Will it offer instruction? Practice? Opportunities for communication? Both of these questions when feasible should be answered before seeking courseware for evaluation.

Once again it should be kept in mind, as the arrows in Figure 4 suggest, that in dealing with the other seven components of learner fit, the goal is first to view the relevant aspects of the operational description through the perspective of each category and then characterize the result in terms of the fit with learner variables, syllabus, or both.

The category of learning style from the Phillips framework overlaps somewhat with the cognitive style considerations mentioned above for the
learner variables component. However, where the former focuses on learner style primarily from the perspective of the learner’s preferences, Phillips’ learning style category is more task oriented and connected to the specific goals of learning as well as learner preferences. This component builds directly on a classification scheme proposed by Kemmis (1977), which distinguishes five general types of learning activities for CALL: recognition, recall, comprehension, experiential learning, and constructive understanding. The first two of these are relatively passive types. In recognition activities the learner need only demonstrate recognition of previously learned material, e.g. noting whether words in a list are spelled correctly or not, while recall activities require only that a learner give a response to previously learned information, such as the definition of a vocabulary item. Comprehension activities, on the other hand, involve recalling and responding to new combinations of learned forms, as in a typical reading comprehension or cloze exercise. Experiential activities are those which involve learning by doing, as in discovering the discourse structure of a language by attempting to reconstruct the scrambled sentences of a paragraph. Finally, activities based on constructive understanding use the computer as an experimental tool to discover new information, as in a conversational simulation like Chatterbox (Phillips 1985).

The component of classroom management in Phillips’ system refers primarily to the grouping of students for a particular CALL activity, but it can be reasonably extended to include other areas where the teacher is involved in controlling or monitoring the learner’s activities. With respect to grouping, software is normally designed to accommodate either one or two learners at the same time, though small group and whole class activities are often possible. Another consideration in classroom management has to do with record keeping. Programs may keep records in a number of areas—scores on exercises or tests, time spent on specific activities, specific errors, etc.

Phillips defines the next component, program focus, as the linguistic objective of the activity. The categories he suggests for this component are discourse/text, syntax, lexis, morphology, and graphology/phonology. As he points out, the hierarchical structure of this classification scheme is such that a program focused at any given level will involve practice at lower levels as well. Sentence level grammar drills, for example, would necessarily involve work at the lexical, morphological, and graphological levels. In using this category in evaluation, a teacher would need to go beyond the basic linguistic category and specify the focus in detail. For instance, if the software had a grammar focus, the
particular morphological or syntactic rules covered would need to be considered.

Moving on to learner focus, this component of Phillips’ system is built around the traditional four skills of listening, speaking, reading, and writing. As Phillips notes, these "are obviously crude and need to be refined in light of particular teaching objectives ... I (P. 33). For evaluation, this refinement would minimally include determining whether the software was aimed primarily at building basic linguistic knowledge through the medium of one of these skill areas (e.g., grammar rules or vocabulary items) or practicing specific sub-skills (e.g., inferring the meaning of a word from context) or both.

Language difficulty is the next of Phillips’ categories to be discussed. He initially suggests a classification scheme with four dimensions of language difficulty: variety, transparency, familiarity, and length. Presumably, texts which are longer and more varied will be more difficult, while those that are familiar and transparent will be less so. Other classification schemes for language difficulty are undoubtedly possible, based perhaps on notions of syntactic complexity, percentage of low frequency vocabulary, paragraph length, and so on. As Phillips points out, it is particularly important to consider this category with respect to control options: in some programs the student may have greater control over this dimension than is possible in a traditional classroom.

The component of program difficulty is the last from Phillips’ framework. It concerns demands placed on the student in interacting with the software that are not strictly linguistic. Specifically, he lists redundancy, timing, and input as subcategories of program difficulty. In addition to these factors which may make a particular program overly challenging to the detriment of the language skill being learned or practiced, it is appropriate to consider the operational complexity of the program as well. For instance, a lack of familiarity with the activity type and the types of control decisions the learner may be allowed can add significantly to the difficulty of operating a program. Where adding these dimensions of difficulty is justified for pedagogical reasons (maintaining interest through variety or allowing a high degree of student control for example), then the evaluator should look for courseware incorporating a tutorial (similar to those often provided for users of word processors, data bases, and other programs whose operation alone is inherently a challenge) or be prepared to spend class time on teaching program operation. One final area of program difficulty involves consistency: screen layout, input conventions (especially use of the Space Bar, Escape, and Enter keys), etc. should be consistent within a single
courseware package to the degree possible so as not to add unnecessarily to the learning burden.

A component implicit in Phillips' framework which is given an explicit role in the present one is the content of the software, which is generally derived from the description of the material component in the operational description. The content ideally fits the needs and interests of the target learners as well as the limitations and pedagogical thrust of the syllabus. The presentation of material that is interesting and engages the attention (sometimes necessarily the repeated attention) of the target learners is compatible with virtually any approach to language teaching and is explicitly called for by some. This is an area where many potentially useful pieces of software fall short, particularly those involving non-contextualized lexical or grammatical exercises if the program itself is not engaging.

There are other dimensions to content as well. One such dimension is the degree to which the language material accurately reflects ordinary, sociolinguistically consistent language (of any appropriate register) rather than language artificially contrived to serve some structural or functional objective at the expense of authenticity. Another, particularly relevant for reading exercises, is the degree of conceptual difficulty associated with the content of a text. An understanding not only of the learners' interests but also of the learners' existing knowledge of a topic is important in evaluating this. A third dimension involves the instructional information provided to the student by certain types of software, particularly those reflecting an explicit learning approach. As in a textbook, such information must be as complete, accurate, and comprehensible as possible, given the practical limitations of the learner and the specific learning task.

Now that the three main components, operational description, learner fit, and teacher fit, have been discussed, they can be combined with the final two parts. Figure 5 lays out the entire evaluation framework, including appropriateness judgments and implementation schemes, which are briefly described below.

**Appropriateness Judgments**

After working through the operational description and getting a sense of teacher and learner fit, the final stage of evaluation is to make appropriateness judgments. As with textbooks, no teacher expects there to be a perfect fit, so the considerations here involve predictions relative to the available options (either other software or non-computer materials). Whether the rest of the evaluation has been done in a strictly analytical fashion or more holistically (see below),
Figure 5: Full Evaluation Framework
there are at least three questions which need to be answered. First, will the courseware be effective? That is, can students actually learn or practice some area of the target language called for by the syllabus? Second, will it be efficient? There are at least two ways to judge potential efficiency. An absolute view involves trying to determine whether spending less time using some other teaching/learning option would give equivalent results. A more relative view is based on determining whether the students are likely to be willing to spend enough additional time using the software to negate any advantage that an alternative activity might have. If the software is inherently more interesting and engaging than an alternative activity, it may still be better in the long run to use that software. Finally, is the program worth the investment? In a department with a limited budget for materials, economic realities may lead to buying several less ideal programs covering a wider range of material than a marginally superior but considerably more expensive package.

Implementation Schemes

One question which is often overlooked in the evaluation process is how the software may be used in ways other than those which the designer outlines in the documentation. In terms of classroom management, for instance, material designed to be used by individual students may be used in pairs or small groups. Baltra (1984), for example, describes how to use an adventure game as a whole class activity by dividing the students into groups and letting each group have access to only a portion of the information necessary to successfully complete the game. As a part of integrating software and syllabus, it is important as well to think about possible preliminary and follow-up activities—in class, as homework, and in the computer lab. It is beyond the scope of this paper to describe the options here in any detail (implementation is given its own module in the comprehensive framework described in the introduction), but thinking beyond the original design is an important aspect of a thorough evaluation.

Using the Framework

One of the goals in developing a framework rather than producing another ad hoc list of evaluation questions was to give an evaluator flexibility with respect to the evaluation format. Consequently, a detailed procedure will not be presented here. A few suggestions, however, will be offered.

First, teachers normally evaluate courseware for one of two reasons: either to decide whether to purchase it for their program or, if it is already available,
whether or not to use it. Given these questions, it is always much easier to decide a particular package is inappropriate than it is to decide that it is appropriate. This means an evaluation procedure should be set up so as to eliminate unlikely candidates as quickly as possible. It is only when writing reviews that an evaluator is required to wade through a package that doesn’t seem appealing and analyze it thoroughly.

Second, while the framework is set up in such a way as to encourage detailed analysis, it is quite workable on a more holistic level. If a teacher has limited time to give to an evaluation, having the framework in front of him or her as a guide will yield a more trustworthy evaluation than simply going by impression.

Even at the grossest level, just addressing the questions "How does it really operate?" "Does it fit my views of language learning?" and "Does it fit my students' needs and interests?" will go a long way toward an informed decision.

Third, as mentioned early in the paper, the framework operates most directly in its present form on relatively simple courseware. The more complex and challenging the programs get, the more complex and challenging evaluating them will be. This is particularly true with respect to authoring systems, where questions about teacher and learner fit may be addressed more honestly from the developmental rather than the evaluative perspective. In evaluating authoring systems, it is best to focus on their flexibility and ease of use, though the operational description part of the framework may be of some help here.

Finally, it should be stressed that, like the evaluation forms and checklists it is designed to replace, even this framework represents a compromise between all the questions a teacher might want to ask and those that someone could answer in a reasonable time. According to an evaluator's needs, then, the framework could be refined, expanded, or abridged without necessarily changing its essential character.

References

Hubbard, P. "Language Teaching Approaches, the Evaluation of CALL Software, and Design Implications." in W. Flint Smith, ed. op. cit., pp. 228-254.
_____. "Two Steps Toward a New Methodology for CALL (Part I)." CALL Digest 4.2 (1988a) pp. 7-8.

Author's Biodata

Philip Hubbard is Lecturer in Linguistics and Coordinator of Summer Programs in English for Foreign Students at Stanford University. He has produced ESL software and published articles on software design and evaluation. He is the Courseware Review Editor for the CALICO Journal.

Author's Address

Philip Hubbard
Linguistics, Bldg. 100
Stanford University
Stanford, CA 94305