Chapter 13 from L. Ducate and N. Arnold (eds) 2006. Calling on CALL: From Theory and Research to New Directions in Foreign Language Teaching. San Marcos: CALICO. Pre-publication copy.

Evaluating CALL Software

PHILIP HUBBARD
Stanford University

1. Introduction

When teachers evaluate a textbook for possible adoption in a language course, they are working in familiar territory. They have used textbooks for years as students and may already have significant experience teaching with them. Although teachers can profit from receiving instruction in how to approach textbook evaluation more analytically (see Byrd, 2001, for example), textbooks are relatively straightforward to evaluate because they tend to have a transparent structure allowing teachers to skim through them to get an overview of the organization and content. Furthermore, teachers can almost always get a full version to review in advance of making a selection.

Not so with software. Although this profile is changing, many current language teachers have limited experience with CALL software from the learners' perspective and may be novices as well using technology for teaching. Unlike textbooks, software structure is often not transparent and can be difficult to "skim" for both content and program operation. Additionally, for commercial materials it may be difficult to get a fully operational version for review. Finally, as Bradin (1999) notes, "language teachers who are not accustomed to looking at CALL software may perceive its purpose very differently than those who are more experienced" (p. 159). All of these factors combine to make CALL software evaluation a unique challenge.

Because evaluation can have various interpretations in CALL and other domains of language teaching, this chapter begins by clarifying the meaning of the term as it used here. Evaluation refers to the process of (a) investigating a piece of CALL software to judge its appropriateness for a given language learning setting, (b) identifying ways it may be effectively implemented in that setting, and (c) assessing its degree of success and determining whether to continue use or to make adjustments in implementation for future use. We may think of these three stages respectively as selection, implementation, and assessment. Historically, software evaluation has been primarily concerned with the first of these stages and the ma-

jority of this chapter is devoted to that area. This is not to say that the other two are less important. However, as we will see, considerations of implementation can and arguably should be an integral part of the selection process. Assessment will be touched on and its value emphasized in the hopes of encouraging readers to engage in further study.

Clarification is also needed for the term "CALL software." It is used here to refer to computer programs and accompanying content that have a recognizable instructional purpose, or a "teaching presence" (Hubbard & Bradin Siskin, 2004, p. 457) and language learning objective. These are sometimes called dedicated CALL or tutorial CALL programs, representing the computer functioning in what Levy (1997, p. 178) refers to as the "tutor" role (as opposed to a "tool" role the computer plays when the program in use is, for example, a web browser, word processor, or email application). Both the degree and quality of the teaching presence vary considerably across CALL software, and, as with a live teacher, the teaching presence can come in different forms. As we will see, one of the keys in evaluation is to determine whether the teaching presence in a piece of software in combination with the content is effective for the given objective.

Tutorial software, also called "courseware," can be of many types, from multiskill commercial products such as *Tell Me More* or *Rosetta Stone* or large-scale single skill compilations such as *Randall's Cyber Listening Lab* to individual language exercises, including those created with CALL-specific authoring templates like *Hot Potatoes*. For those new to the field wanting a clearer understanding of the nature and extensive range and variability of tutorial software across a number of languages, it is instructive to begin with a visit to the *CALICO Review* web site. Although our focus in this chapter is on tutorial software, it is worth pointing out that many of the considerations described here are relevant for other forms of evaluation. These include the evaluation of CALL activities or tasks (for instance, web quests or computer-mediated communication activities) that either do not involve tutorial software or represent blends of tutorial and tool-oriented applications.

In the next section, the different purposes of evaluation, such as selection of software for an individual course or for a self-access lab, are touched upon. Then, three major approaches to CALL software evaluation are described and compared: checklists, methodological frameworks, and SLA research-based approaches. Following the section on the three major approaches to evaluation is an extended example of a methodological framework similar to the one that informs the *CALICO Review*, followed by a discussion of implementation considerations. The chapter concludes with suggestions for continuing to evaluate software appropriateness and effectiveness during and after student use, along with some final comments.

2. Purposes of Evaluation

2.1 Selection for a Course

The most common reason for doing an evaluation is for a teacher to select appro-

priate software for his or her own class. In this situation, there is a lot of known information that can be brought to bear on the evaluation process. Such information includes (a) an understanding of the technical infrastructure of the institution or the computing hardware and software available to the students if they are using their own equipment, (b) relevant data about other course materials, the student characteristics, and the structure and specific objectives of the course, and (c) the teacher's/evaluator's assumptions about how language is learned. Besides the classroom setting where many of the preceding factors are known, teachers may also evaluate software for use beyond their own courses.

2.2 Selection for Self-access or Other Instructors' Use

In some situations, teachers or other language professionals may be asked to recommend software selections for a self-access lab or for an entire language program. In this case, the same considerations of technology infrastructure will presumably be taken into account, but the information about student characteristics, course objectives and materials, and teacher assumptions may be less readily available. In addition, there can be a great deal of variability in student and course characteristics that can make the selection process more challenging. Evaluators in this circumstance would do well to begin with some investigation of these factors rather than simply relying on their own assumptions and biases.

2.3 Reviews

Reviews differ from other forms of evaluation in that they typically focus on the software itself rather than on the environment in which the software will be used. Published reviews such as those found in the *CALICO Journal* and on the *CALICO Review* web site are aimed at a broad audience of potentially interested parties. As a form of evaluation, a review is an important source of information that others can use both in making the initial identification of possible candidates and in informing their own evaluations.

3. Three Approaches to Evaluation

Levy and Stockwell (in press) have identified three major types of CALL software evaluation: evaluation driven by checklists or forms, evaluation guided by methodological frameworks for language teaching, and evaluation linked to second language acquisition (SLA) theory and research-based criteria. Each of these is discussed briefly below.[1]

3.1 Checklists

Checklists have been present from the earliest stages of CALL and remain widespread. Typically, a checklist presents a series of questions or categories for judgment and the evaluator is expected to make a response based on information gathered through the reviewing process. Many checklists simply ask for a yes/no indication or a response along a Likert scale. Others, despite the "checklist" label,

also include space for open-ended commentary following specific prompts. Published checklists have been criticized for a number of reasons, including focusing too heavily on technology at the expense of pedagogy and for being biased and restrictive (Hubbard, 1988). However, Susser (2001) provides a rebuttal to many of those criticisms and builds a convincing case for the value of CALL evaluation checklists. The thrust of his argument is that the problem is not with the concept of checklists but rather with particular instantiations of them.

There are numerous examples of software checklists available online for those interested in pursuing this option. The International Society for Technology in Education (ISTE) has a highly regarded general software evaluation form for educators available for download. An example of a checklist that demonstrates a strong consideration of language learning research findings is one housed on the web site of the National Foreign Language Resource Center in Hawaii that was developed specifically for multimedia language learning software. A checklist by a group of students at Rice University (Garrett, Nimri, & Peterson, n.d.) built on a methodological framework is also available online.

In the end, as Susser (2001) notes, checklists do not have to be accepted as is but can be adapted and updated for particular purposes. They have the capacity to provide teachers with a useful tool for recognizing the variety of elements that make up a software application and for triggering reflection on some of their own assumptions about CALL. Finally, among the many possible sources for items on a checklist are methodological frameworks and SLA research, as discussed below.

3.2 Methodological Frameworks

Methodological frameworks are compatible with some checklists but differ in two significant ways. First, methodological frameworks attempt to be largely descriptive rather than judgmental in their form. Second, they attempt fundamentally to link with the language teaching and learning considerations that take place outside of technology. As noted in Hubbard (1988),

> The framework approach to courseware evaluation is different from others. … 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 a tool through which an evaluator can create his or her own questions or develop some other evaluation scheme. (p. 52)

Until the mid-1980s, evaluation had largely been conceptualized in terms of checklists and procedures borrowed from general education and was lacking an appropriate language learning focus. But in 1985, Phillips offered a framework more explicitly linked to language teaching methodology. It included categories for the CALL program types of its era but also described dimensions such as language difficulty, learner focus (i.e., skill area—listening, speaking, reading, and writing), and language focus (i.e., lexis, grammar, and discourse) that were important to the language learning character of the program.

Hubbard (1988) expanded Phillips' system and integrated it with one developed independently by Richards and Rodgers (1982) for describing and analyzing language teaching methods. Richards and Rodgers (themselves building on an earlier framework by Edward Anthony) characterized language teaching methods in terms of three descriptive categories: (a) *approach*, or the underlying theories of linguistics and language learning assumed by the method; (b) *design*, consistent with the assumptions of the approach and including the syllabus model, general and specific objectives of the method, and the roles of the students, teacher, and materials; and (c) *procedure*, or the classroom techniques and activities through which the design is realized. Hubbard (1988) adapted the approach, design, and procedure constructs into categories describing the key elements of evaluation and renamed them *teacher fit, learner fit,* and *operational description*, respectively.

The resulting framework became the evaluation module in a proposed comprehensive methodological framework that also included modules for courseware development and implementation (Hubbard, 1996). A version of this framework remains at the core of the review procedure for the *CALICO Journal* (for details, see the Appendix to this chapter; Burston, 2003).

3.3 SLA-based Approaches

Given that teaching languages with software *is* a form of language teaching, another reasonable procedure for developing software evaluation rubrics is to build on recommendations from theory or research in instructed SLA. Ultimately, we might expect to have definitive SLA results specifically from research on learning with software, but to date there has not been a sufficiently established base for such results. Consequently, this approach takes findings from non-CALL domains and interprets them in the CALL context.

An early attempt in this direction was Underwood (1984), who presented a case for a communicative approach to CALL based on generalizations from research and communicative theory of that period. His 13 points characterizing communicative CALL became a de facto evaluation rubric. Egbert and Hanson-Smith (1999) structured the chapters in an edited volume on CALL around eight generalizations for optimal language learning environments, again providing content for a research-based evaluation scheme, although their work was not specifically aimed at evaluation.

The most ambitious project in this vein to date is represented by the work of Carol Chapelle in the field she has dubbed CASLA—computer applications in second language acquisition—which includes not only CALL but also computer-based language testing and computer-based SLA research. Although parts of the model were developed in earlier articles, the work comes together in Chapelle's 2001 book, which represents a significant advance for (a) its characterization of evaluation on the basis of principles and (b) its specific SLA-based criteria. With respect to the first point, Chapelle offers a set of five principles for evaluating CALL summarized as follows:

1. CALL evaluation is situation-specific;
2. CALL should be evaluated both judgmentally and empirically;
3. CALL evaluation criteria should come from instructed SLA theory and research;
4. the criteria should be applied relative to the purpose of the CALL task; and
5. the central consideration should be language learning potential.

In line with the preceding principles, Chapelle proposes a set of six general evaluation criteria useful in determining the appropriateness of a given CALL task for supporting language acquisition. Note that these criteria are relevant for "both the aspects of the task defined by the software and those defined by the teacher" (Chapelle, 2001, p. 58). These criteria appear initially in Chapelle (2001) and are reprised in a recent evaluation study by Jamieson, Chapelle, and Preiss (2005, p. 94).

1. *Language learning potential:* The degree of opportunity present for beneficial focus on form;
2. *Learner fit:* The amount of opportunity for engagement with language under appropriate conditions given learner characteristics;
3. *Meaning focus:* The extent to which learners' attention is directed toward the meaning of the language;
4. *Authenticity:* The degree of correspondence between the learning activity and target language activities of interest to learners out of the classroom;
5. *Positive Impact:* The positive effects of the CALL activity on those who participate in it; and
6. *Practicality:* The adequacy of resources to support the use of the CALL activity.

Jamieson, Chapelle, and Preiss (2004) show how these criteria can be operationalized for a judgmental analysis of a major software project, *Longman English Online* (*LEO*).[2] In a follow-up study (Jamieson, Chapelle, & Preiss, 2005), they again build on these criteria to create a rubric for evaluating *LEO* empirically, eliciting data from the software developers, a teacher using the software, and a set of student users. Table 1 shows an example of their entry for "language learning potential."

It is worth noting that Chapelle's framework, though quite different in structure and in underlying assumptions, is in some respects compatible with the methodological framework and checklist approaches described earlier. For instance, Chapelle's concept of learner fit can be related to that of Hubbard (1988), and most of her other criteria are representative of a task-based, interactionist language teaching approach that is likely to provide a good "teacher fit" for many current language instructors, especially those who have been recently trained in such an approach. Finally, as Table 1 illustrates, the result of an SLA-based approach can be a principled checklist.

This section has outlined approaches to evaluation based on checklists, methodological frameworks, and SLA research. While all three have their merits, the re-

mainder of this chapter will focus on presenting the methodological framework in more detail since it is the most neutral in terms of language teaching approach.

Table 1
Example of CALL Criteria and 0perationalization from the Chapelle (2001) Framework*

Criteria	Operationalizations	Desired responses to support claims for quality
Language learning potential • Sufficient opportunity for beneficial focus on form	• Will the grammar, vocabulary, and pronunciation that was studied during the week be remembered?	• Yes
	• Were the explanations clear?	• Yes
	• Were there enough exercises?	• Yes
	• Will the students' English improve as a result of *LEO 3*?	• Yes
	• Will the students' quiz scores indicate mastery of the material?	• Yes

*Excerpted from Table 2 of Jamieson, Chapelle, and Preiss (2005, p. 99)

4. A General Evaluation Framework

The following description is based on the assumption that evaluation is being done for the most common purpose, namely a single teacher selecting software for integration into a particular course. However, the framework outlined in this section can be readily extended to any of the domains mentioned previously, that is, selection for self-access or other instructors or evaluation for a formal review (see Appendix). Note also that software packages are often complex, including a number of different types of presentations, activities, exercises, and quizzes, so in a thorough evaluation it will be necessary to cover examples of each type.

4.1 Rationale

Two of the articles noted above (Hubbard 1988, 1996) outlined a methodological framework for CALL combining elements of development, evaluation, and implementation viewed primarily through the lens of language-teaching approach and design considerations. The description that follows remains true to the assumption that language teaching and learning judgments are at the core of CALL software evaluation rather than, or in addition to, principles from instructional design or other areas of education. This is both because of the unique nature of language learning and because many teachers performing these evaluations do not have direct experience in these other areas. Also, by linking CALL software evaluation to language-teaching methodology, the connections necessary for integration can be much more readily made.

This chapter focuses on a methodological framework rather than the alternatives primarily because of the methodological framework's descriptive and more comprehensive nature. Theory and research-based approaches such as Chapelle's (2001) are by their nature prescriptive, at least in terms of approach, because they

are based on a particular conception of language teaching and learning (even one as well established as that underlying Chapelle's). A checklist procedure is avoided for the same reason, not because of any intrinsic limitations of checklists. In fact, it is claimed here that both checklists and SLA-based evaluation criteria can be largely accommodated by a descriptive methodological framework. Further, a type of checklist—a list of considerations—can be generated directly from the categories and elements in the following methodological framework.

4.2 Preliminaries—Identification of Potential Software

Before beginning any evaluation, we need to identify candidates to evaluate. In most cases, these will probably be programs designed for language learning. However, it is worth pointing out that teachers have had success with programs designed for native speakers of the target language; in fact, many of the programs in the TESOL CALL Interest Section list mentioned below (Healey & Johnson, 2005) fall into this category.

The first step is to have a basic understanding of what characteristics to look for in candidates. While a formal needs analysis is ideal, even a brief look at your existing course structure will help in identifying programs with potential. Healey and Johnson (1997/1998) offer a useful set of guiding questions for this step.

1. Who are the users you are targeting?
2. What are the goals of the students you are targeting?
3. What setting will the software be used in: independent study lab with no teacher available, lab associated with a class, a teacher-led class with one or a few computers?
4. How much do the teachers/lab assistants who will work with the students know?
5. What do you have now in the way of hardware and technical assistance?
6. How much money do you have to spend?

There are many sources of information about language learning software. For ESL/EFL, the most comprehensive source is the TESOL CALL Interest Section Software list (Healey & Johnson, 2005). For all languages, two useful sources are the *CALICO Review*, both the current issue and archives, and the database at the National Foreign Language Resource Center at Hawaii. Commercial vendors (and even academic authors distributing their software for free) also have useful information on their products, but it is a good idea whenever possible to supplement the information they provide with independent reviews or comments from other teachers who have used the programs.

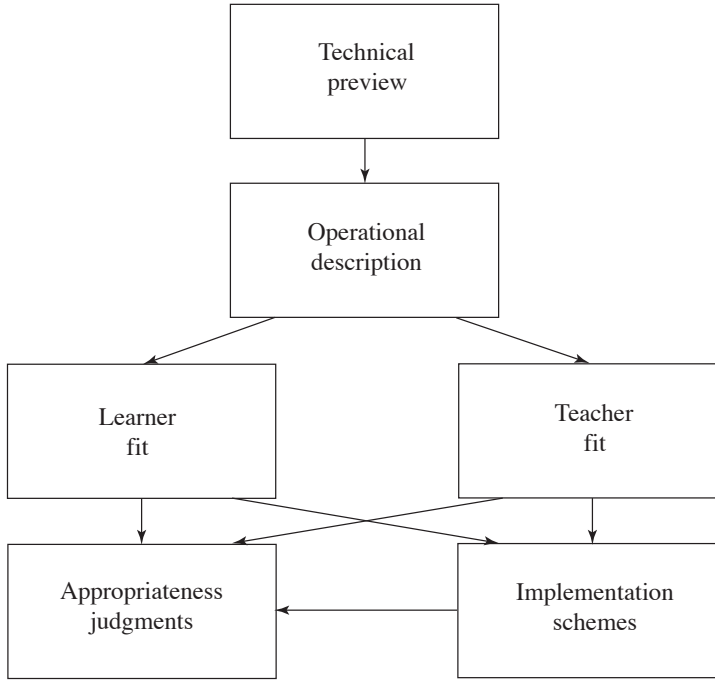
Robb and Susser (2000) conducted a survey of language teachers through several listservs to explore their software selection process, focusing on 11 possible "sources of information potentially affecting purchase decisions" (p. 45). Some selected results from their 51 respondents are instructive. Eighty-one percent claimed to have obtained the information needed for selection by getting a full or demonstration version of the program and evaluating it themselves, while 37% relied on information from a colleague, 31% on reviews from the literature and

only 25% claimed to have used a checklist. Interestingly, when asked about the source of information for software they continued to use on a regular basis (rather than just initially purchasing software), the largest group (78%) reported relying on a recommendation from a colleague. This and other results from their study, despite being from a limited, nonrandom sample, are consistent with their findings that many practicing teachers use a variety of information sources (not just a checklist-based review) to select software and that colleagues are a particularly good source of relatively reliable information.

4.3 Overview of Basic Structure

We begin by assuming that there has been a "preevaluation" and that the software under scrutiny is a reasonable candidate for ultimate selection. Figure 1 presents the basic structure of the evaluation framework, adapted from Hubbard (1988, 1996) and Burston (2003).

Figure 1
Software Evaluation Framework: Core Components



At this level, even without further analysis, it reflects a simple evaluation procedure embodying the following stages:

1. *Technical preview.* Make sure that the software will run the way you want it to on the equipment that you or the students have available.
2. *Operational description.* Go through the main parts of the software as a cooperative user (you can try to be less cooperative later). Get an understand-

ing of the flow of lessons and items within them *before* making judgments. You can record your first impressions but try to withhold judgment until you understand how the software actually operates.

3. *Teacher fit*. Try to infer the language teaching approach that the software reflects (which may be different from what the designers claim) and determine the degree to which it is compatible or incompatible with your own.
4. *Learner fit*. Note how well the content, skills, and language level correspond to your students' needs, especially as determined by the objectives in the course syllabus. Note also how well the software fits the students' interests and preferred learning styles.
5. *Implementation schemes*. Reflect on how the software might be integrated into the course or a curriculum, including what students will need to know in order to use it effectively and how much time that process will take.
6. *Appropriateness judgments*. Ultimately, make a decision to use or not, based on the quality and degree of teacher fit and learner fit, along with considerations of the costs and benefits of implementation. Keep in mind that no software will be a perfect fit. It may be helpful to think of judging a program's teaching presence the way you would judge a human teacher. Ultimately, one or more humans created the materials and structured the program's actions for learning purposes. Do you want that person or persons teaching your students with the material they provided?

4.4 Technical Considerations

Technical considerations are of several types. The most basic one is: Will it run on the machines the students will be using? The primary split here continues to be between Microsoft Windows-based PCs and Apple Macintoshes (though Linux seems to be gaining some ground). Some software runs on all platforms, particularly web-based applications, and some on just one. However, as software is developed for PDAs and even cell phones, this question may become even more complex. The operating system version, memory requirements, and other hardware and software issues are often significant as well. If online materials are being used, how accessible are they? Can students use them only in a lab, or are they available on a larger institutional network or even the web? Additionally, for online materials there are issues of bandwidth and server access—the speed at which data is transferred and the impact of multiple users tapping into a single application. This is a particularly important point when using audio and especially video files, which may be delayed or degraded over slow connections.

A description of all the technical issues is well beyond the scope of this chapter. Evaluators who have concerns about their ability to make the preliminary judgments at the technical level are encouraged to seek assistance from others who are more knowledgeable, especially before making purchase decisions.

4.5 Operational Description

The operational description is a review of the components of the software and

how they operate on their own or are controlled by the user. It is meant to be an objective description that can then be used to feed the judgmental aspects of the framework. In the original framework, the operational description was presented as a set of more or less independent central and peripheral descriptive categories, and this remains a useful classification. However, we will briefly consider an alternative conceptualization for some of its elements at the end of this subsection.

The peripheral categories include any accompanying text, documentation, tutorial (on how to use the software), record keeping features outside of the main program, and any other utilities such as teacher authoring capabilities. The central categories include the general *activity type* (e.g., game, quiz, text reconstruction, exploration, or simulation) and the *presentational scheme*, which describes the way a CALL activity is presented to the learners. The reason for this distinction is that a given activity type may have a number of diverse presentational schemes. For example, the text-reconstruction activity type would likely have quite different presentational schemes for hangman, a standard cloze exercise, and a scrambled sentence activity.

Presentational schemes are defined by a number of subcategories.

1. The *screen layout* or *interface* is concerned with all aspects of the basic appearance on screen, including fonts, color schemes, controls, as well as presence, placement, and quality of graphics, video and audio. This is a major area of investigation in the broader field of human computer interaction (HCI) and may involve cultural factors in addition to the more obvious cognitive and esthetic aspects.
2. *Timing* is a relevant category for some software, for instance, by limiting the time that content material or a prompt appears on the screen, by limiting the time allowed for a response, or by recording the time it takes for a student to perform some action.
3. The c*ontrol options* category describes what is under learner vs. program control as well as the physical nature of those controls. For example, does the learner have the ability to go to any desired lesson through a menu, or does the program require the student to complete one lesson before moving on to another in a predetermined order? Can the learner call up text support for an audio or video exercise, or is it always present (see Chapter 5 for a detailed discussion of listening comprehension in CALL)? This is an arena of some debate within CALL and CAI (computer-assisted instruction) in general. A study by Boling and Soo (1999) for instance, found that novice language learners tend to prefer more structured CALL software while advanced learners are more comfortable with taking control themselves.
4. *User input* (a category missing from earlier versions of the framework) characterizes how the learner responds to implicit or explicit prompts from the program (e.g., speaking, typing, clicking a button or hotspot, etc.).
5. *Input judging* describes the program's procedure for handling user input, which can involve such actions as recording a mouse click, various types of pattern matching, speech analysis, or linguistic parsing.

6. *Feedback* is provided to the user by the program as the result of the input judging. This is a key part of the description of the presentational scheme because there are a number of options, some of which clearly represent a more active teaching presence than others. Feedback can be either implicit (as when an incorrect answer simply disappears as a choice when it is selected) or explicit. For a typical quiz or practice exercise, feedback can simply indicate a correct or incorrect response, or it can provide additional information in the form of hints or explanations. For other types of programs, such as simulations, feedback can take other forms (e.g., in a simulated dialogue a character in the program might respond orally to the student's input or perform some requested action). Feedback may also be cumulative, as when the program saves scores and other performance data for the student to review.
7. *Help options* represent the final element to consider in the presentational scheme. In addition to a description of the content of the help, the key points here are whether any assistance that is provided is contextualized and targeted for a given item rather than being global and whether help is available at all times or only under certain conditions.

As noted above, the operational description in previous versions of this framework did not directly portray the connections among the various elements, especially those in the presentational scheme (for details, see Hubbard, 1988). An alternative way to conceive of these elements is to model their operation more dynamically at the microlevel as an *interactional sequence*: a set of exchanges between a program and a user on a single point or topic, such as a question in a tutorial exercise. An interactional sequence involves one or more turns by each party (the computer and the user) prior to shifting the topic (e.g., moving to the next question or item) or ending the interaction.

Here is an example of a typical interactional sequence: Assume a novice ESL learner is using a program to learn vocabulary through picture identification. In this lesson, the focus is on verbs associated with particular animals. The activity is set up on the screen as four pictures: a fish, a lion, a snake, and a bird. These represent the response domain (what the learner can click on).

Computer:	Which of these can fly? [prompt]
Learner:	(clicks on the fish) [input]
Computer:	Sorry, that's not right. Fish can't fly. Fish swim.
	(fish swims off screen) [feedback]
	Which of these can fly? [prompt]
Learner:	(clicks on the bird) [input]
Computer:	That's right. Birds can fly. (bird flies off screen) [feedback]

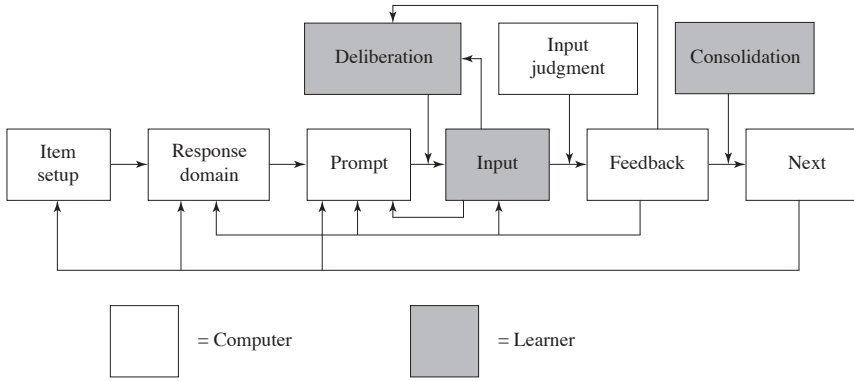
(the screen refreshes and a new set of pictures appears …)

Hubbard (2001) offers an initial model for such sequences that covers three common tutorial modes: presentation, exploration, and interrogation. In the preceding example, the sequence is in the interrogation mode because the computer is doing the asking, prompting the student for a response, and then providing

feedback on the accuracy of that response. If we consider the same type of example, namely teaching vocabulary through pictures, we can see how the other two modes differ from interrogation. In presentation mode, the program would simply provide information in the target language, for example, highlighting each of the pictures (fish, lion, snake, and bird) one by one and playing an appropriate description ("This is a bird. A bird can fly"). As with an audio or video tape recording, the student's interaction with the program would be limited to such actions as pausing, repeating, adjusting the volume, and so on. In exploration mode, the learner is the one who does the asking, in this case by clicking on pictures as desired to get information about them, with the option of ignoring those he or she already knows.

Although a detailed explanation of interactional sequences is beyond the scope of this chapter, there are two concepts in this model that have a bearing on both the selection and the implementation process. The interactional sequence model for possible paths in interrogation mode is shown in Figure 2, though of course not all paths will be open to every item—the richness of the options in an interactional sequence is determined by the program designer.

Figure 2
Interactional Sequence Model for Interrogation Mode (Hubbard, 2001)



Note: The white boxes indicate what the computer does, and the gray boxes what the learner does.

The concepts of importance here are *deliberation* and *consolidation*. Deliberation refers to the cognitive operations by the learner prior to taking a physical action (inputting something: a mouse click, text, speech, etc.). Deliberation is the largely invisible activity of reflecting and accessing the relevant information to either answer the prompt correctly, make an educated or random guess, ask for help or a hint, or give up. It is fed not only by all that precedes the initial input but also by the feedback from the computer after the first "round" of the interactional sequence. Consolidation is a similar cognitive activity that involves taking whatever bits of information were "learned" from that interaction and taking a moment to reflect on them in the hopes of promoting retention and integration with existing knowledge.

How much a given program supports (or does not support) deliberation and consolidation is thus another potential consideration in the evaluation process. The degree to which learner training (see below) can encourage deliberation and consolidation and the possible cost in time and resources of providing that training are additional points to ponder.

4.6 Teacher Fit

Teacher fit represents considerations largely at the level of Richards and Rodgers' (1982) characterization of *approach*. This begins with the evaluator's assumptions, ideally supported by theory and research, about two areas. The first of these concerns assumptions about the nature of language, including issues such as the relationship between language and communication and the relationship between language and culture. The second set of assumptions (presumably compatible with the first) is about how languages are learned. Together these form the evaluator's *language-teaching approach*. A further set of considerations emerges from the evaluator's understanding of the capacities of the *computer as a delivery system* for both content and pedagogy. Combined with the language teaching approach, these considerations yield *approach-based evaluation criteria*.

In practice, these criteria can remain at a somewhat intuitive level (assuming we can recognize when a language learning activity is at odds with our approach), or they can be operationalized into a checklist or some other form. For example, Hubbard (1988, p. 63) gives the following partial set of evaluation criteria for what were referred to in that work as "explicit learning" approaches:

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 alternative correct answers within a given context;
4. offers the option of explanations for why correct answers are correct; and
5. anticipates incorrect answers and offers explanations for why they are incorrect.

Regardless of whether they are used directly in the evaluation process, it is an instructive exercise for any evaluator to experiment with producing some approach-based evaluation criteria since this necessitates reflecting on one's own beliefs about language, language learning, and the capacity of the computer to support language learning as well as the foundation on which those beliefs rest.

4.7 Learner Fit

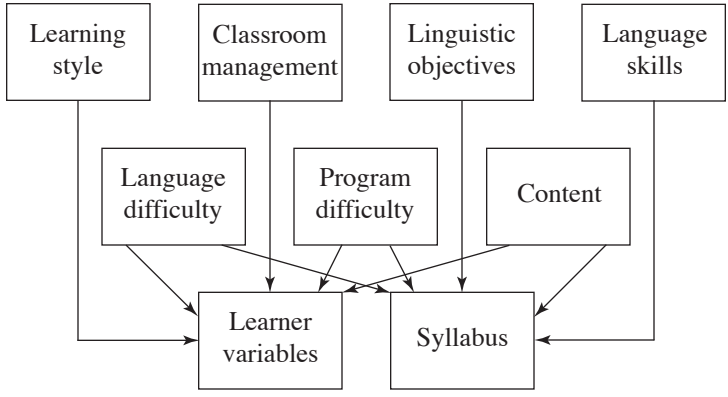
Learner fit covers many of the same topics as Richards and Rodgers' (1982) concept of *design*. The two central areas in which we are looking for compatibility are *learner variables* (e.g., age, native language, proficiency level, sex, learner needs, and learner interests) and the *syllabus*. Each of these variables is fed by a number of other elements, some of which influence both. It should be noted that learner variables are by their nature individual and in some cases not obvi-

ous to the teacher. However, if a class or group of students within a class can be identified as having similar attributes, these variables become important in the evaluation process. The concepts of *learning style* (taken here broadly to subsume cognitive style, preferred learning strategies, and motivational orientations) and *classroom management* (such as whether the software is to be used individually, in pairs or groups, or as a whole class activity and the degree to which the learners must be monitored) represent considerations associated with learner variables; in particular, age and native language and culture. The *linguistic objectives* ("program focus" in earlier versions of the framework and in Phillips (1985), are discourse/text, syntax, lexis, morphology, and phonology/graphology: these are part of learner fit because they are related to the syllabus, as are the *language skills* targeted by the software (previously "learner focus"), such as listening, speaking, and so forth.

The three remaining elements are connected to both learner variables and the syllabus. The idea of *language difficulty* represents the level of linguistic challenge along several dimensions, such as grammar, lexicon, and speed of presentation and clarity of pronunciation (for audio or video). The notion of *program difficulty* has to do with the learning curve to operate the program (because of inherent complexity or technical design flaws) and, in the case of games, the deliberate level of challenge the user is faced with. In both cases, the time spent in pursuits not directly tied to language learning as well as the potential for frustration are factors that weigh against the likely language-learning achievements and any positive impacts on motivation. Finally, *content* is an issue for both the syllabus and learner variables. The content should be considered with respect to its consistency with course objectives (e.g., if the target culture is an element of the syllabus, the cultural content should be authentic and appropriate) and with respect to student interests and existing knowledge.

Figure 3 (a variant of Hubbard (1988, 1996) shows the relationships among the elements of learner fit.

Figure 3
Elements of Learner Fit



4.8 Appropriateness Judgments

Teacher fit and learner fit considerations combine to yield judgments of the appropriateness of a piece of CALL software for a given setting. Although the process of getting to this point appears linear in Figure 1 above, decisions to reject a program can be made at any point: for instance, if either teacher or learner fit is noted early on to be poor, there is no need to continue. The result of this process is a decision of whether or not to recommend purchase of the software, or, if it is already purchased or available for free, whether or not to use it. However, a teacher evaluating a promising piece of software must determine not just whether but also *how* and *when* to use it: this is the domain of implementation.

5. Implementation Schemes

5.1 General Considerations

Although implementation logically occurs after the selection stage in evaluation, looking at the way in which a piece of CALL software may be implemented is important for determining how to use it effectively and may influence whether or not it is purchased (see also Chapelle's [2001] criterion of practicality). In the common situation where an institution has limited funds for procuring software, implementation issues may clearly favor one program over another when the two may otherwise be equally good candidates in terms of teacher and learner fit. The implementation module in Hubbard (1996) provides several points of consideration that converge on the development of schemes for learner use. These include accessibility, preparatory and follow-up activities (especially if linked to other course materials such as a textbook), and a number of teacher-controlled variables such as classroom management, site monitoring, teacher program control (for programs that allow such settings), access to student records, and teacher authoring possibilities.

5.2 Learner Training

The teacher's job is not complete when the software has been selected, procured, and integrated into the syllabus. In most cases, students need time and training to learn how to use the software effectively from a pedagogical as well as a technical perspective. In particular, they need to learn how to connect their actions with the software to desired language learning objectives. For instance, they need to be guided to understand that the primary purpose of a computer reading or listening lesson is not to answer the comprehension questions correctly but rather to engage with the language and content to improve their reading or listening proficiency—comprehension questions are just the most visible part of that process.

Hubbard (2004) argues that learner training should be a significant part of CALL software implementation and proposes five guiding principles for CALL learner training:

1. experience CALL yourself from the learner's perspective;
2. provide learners with some teacher training so that they can make better decisions when working independently;

3. employ a cyclical approach, making training ongoing rather than relying on one-time training sessions when the software is first introduced;
4. use collaborative debriefings to encourage students to reflect on their learning process after using the software and to promote finding out about effective procedures from one another; and
5. teach general exploitation strategies so that they can take greater control of the software and adapt it in ways that go beyond the designer's vision.

Kolaitis, Mahoney, Pomann, and Hubbard (in press) report on a project that implemented these learner-training principles into an ESL program at a community college. They found that while some of the learner training principles were helpful, others, such as giving learners teacher training and finding time for collaborative debriefings, proved much more of a challenge for teachers. They noted, however, that developing the materials and procedures for training their students allowed them to have a clearer view of the need to link software use to learning objectives and to teach specific CALL strategies to promote that linkage.

6. Evaluating Student Outcomes

A final area of the evaluation process that needs to be touched upon is determining the degree to which the software is used and the manner in which it is used have been successful. This assessment process helps the teacher decide whether to use the software in the future, and, if so, whether to use it in the same way or differently. It also adds to the teacher's general understanding of what students do with the software, which can influence future evaluations and implementation decisions. To this end, Chapelle (2001) provides a set of questions for determining the results of student use, empirically reflecting the six criteria presented previously for judgmental evaluation. For example, for the criterion of learner fit, she offers the following: "What evidence suggests that the targeted linguistic forms are at an appropriate level of difficulty for the learners? What evidence suggests that the task is appropriate to learners' individual characteristics (e.g., age, learning style, computer experience)?" (p. 68). A more elaborated version appears in Jamieson, Chapelle, and Preiss (2005).

Although important in principle, this sort of evaluation can be quite challenging and time consuming to accomplish well in practice. Even some empirical information is better than none, however, so the use of one or more of the following methods is highly recommended. It should also be noted that this kind of empirical study with students can be done at a "pilot" level during the selection stage if a trial version of the software is available. In fact, Robb and Susser (2001) report that 56% of their survey respondents during the selection process "obtained a copy/demo and had some students try it" while 52% "used it under class conditions" (p. 46).

6.1 Observation

The most direct way to get information on whether the software is having a positive effect on learning is by watching the students as they use it. In a lab situation,

particularly when dealing with software that is new to the student, the teacher can walk around, take note of how students are moving through the software, and interact with them as they are interacting with the software. Information gleaned in this manner can be used both to evaluate the software and to inform ongoing learner training.

6.2 Tracking Systems

Perhaps the best way to get objective information on student use is either to select software that includes tracking of student actions or to employ a screen capture device (e.g., *Lotus Screen Cam*) that will record the changes in the student display. Depending on the type of tracking system used and the nature of the data collected, this can allow for either a superficial overview, for example, student quiz scores or time on the computer (not necessarily the same as time on task) or a data set that is rich in detail but may be time consuming to analyze. Other things being equal, however, the presence of a tracking system in software is seen as a positive feature.

6.3 Student Surveys

Another approach to gathering information on student perceptions of success or failure with the software is to ask them by using a survey or questionnaire. While such information can be valuable, there are two concerns. First, if students know their responses are tied to a grade or other assessment, or if they believe (even erroneously) that this is the case, the results will be compromised. Thus, it can be important to ensure anonymity if feasible. Second, even when students are trying to be completely honest, their reports may not correspond to their actions. Fischer (2004) reported on a study of French reading software in which the student accounts of their use of program features were quite different from what was observed in the objective data in the tracking logs. If surveys are to be used, then it is advisable to administer them either during or immediately after completion of a CALL activity to tap into fresh memories as much as possible.

6.4 Pre- and Posttesting

Evaluating student outcomes is a form of research, especially when it is done with software that is untried for a particular setting. Certain types of CALL instruction, particularly those which can be assessed with some degree of validity with discrete-point tests such as vocabulary development, may be empirically evaluated using a pre- and posttest regime. Note that while this may give useful information on the outcome, it does not provide the data about the learning process that most of the other options do. It does, however, often have strong face validity with students and school administrations, especially when results are positive.

6.5 Student Journals

Kolaitis et al. (in press) report success having students keep a “CALL journal” in

which they include not only the time and description of the material worked on but also reflections on why they got certain answers wrong in exercises. Although this is mainly done for the students' benefit, this kind of journal also provides teachers with useful information on how their students are progressing and using the software. Note, however, that like questionnaires, the data in student journals may not be fully reliable and should be interpreted accordingly.

7. Conclusion

Software evaluation remains an important area of CALL and there are indications its role may be increasing, particularly in the domain of empirical evaluation. Reeder, Heift, Roche, Tabyanian, Schlickau, and Gölz (2004) have proposed a learner-centered theory to support empirical "E/Valuation" (p. 56) built on relationships among learner variables, learner objectives, and both constructivist and instructivist language-teaching methods. Their paper describes a collaboration among the authors' four institutions and outlines an ambitious plan for a four-nation study of multimedia software, the development of an online technical manual of reviews and E/Valuation techniques, and the founding of an International Institute for Language Learning Software Evaluation.

However, for most language teachers software evaluation will remain primarily a judgmental process, ideally with some empirical follow up of the types described in the previous section. Even at the judgmental level, though, thorough software evaluation of the type often mandated by published checklists and procedures is a time-demanding process that will be impractical for many classroom teachers. The question remains then, which evaluation procedure to select. Robb and Susser (2000) suggest an answer.

> The vendor who explains everything, the colleague who remembers everything, the checklist that covers everything, and the framework that suggests everything do not exist and if they did, would probably be impossible to use. Consequently, software selection is still very much an art honed by experience (p. 49).

Reflecting on the material and procedures provided in this chapter is an important first step in mastering that art, but the challenge remains for the individual to gain the experience needed to determine a compatible CALL software evaluation procedure that is practical and consistently yields reliable results.

Questions for Reflection

1. Recall the three major types of evaluations: checklists, methodological frameworks, and SLA-based approaches. If you had to evaluate a piece of software today for possible use in your class, which approach, or combination of approaches, would you use and why?
2. Visit the *CALICO Review* site at www.calico.org/CALICO_Review. Select a review that is relevant to your current or future teaching and read it critically. Think about (a) what useful information is provided, (b) what support the reviewer offered for the judgments made, and (c) what information was not provided that you would have found helpful. If possible, compare your own experiences working with this program with the reviewer's opinion.
3. The methodological framework offered in this chapter requires a judgment of teacher fit. What is your language teaching approach, and what approach-based software evaluation criteria can you generate from it?
4. Locate a piece of language learning software and find one activity in it. Using the model in Figure 2, see whether you can describe the interactional sequence of an item. What could you tell students about using the software that would help their deliberation and consolidation?
5. Some software is designed by publishers to go with a particular textbook, while other software is independent of any text. What are some of the advantages and disadvantages of each?
6. If you are currently teaching or have recently taught a language class, think about the kinds of CALL software support materials that could either be integrated into your class or made available to students for supplementary work. How would you go about locating such materials once you identify the need?
7. If you have never learned a language using tutorial software, try to find some at your institution or on the web and experience CALL from the learner's perspective. Reflect on how your experience as a learner can inform evaluation decisions that you make as a teacher of a language you know well.

Notes

[1] For a more comprehensive contemporary treatment of these three approaches covering not only software use but also other CALL tasks, see the evaluation chapter in Levy and Stockwell (in press).

[2]As Jamieson, Chapelle, and Preiss (2005) note, the *Longman English Online* product was dropped by Longman and much of its content has been released on CD-ROM as *Longman English Interactive*.

References

Boling, E., & Soo, K.-S. (1999). CALL issues: Designing CALL software. In J. Egbert & E. Hanson-Smith (Eds.), *CALL environments: Research, practice, and critical issues* (pp. 442-457). Alexandria, VA: Teachers of English to Speakers of Other Languages.

Bradin, C. (1999). CALL issues: Instructional aspects of software evaluation. In J. Egbert and E. Hanson-Smith (Eds.), *CALL environments: Research, practice, and critical issues* (pp. 159-175). Alexandria, VA: Teachers of English to Speakers of Other Languages.

Burston, J. (2003). Software selection: A primer on sources and evaluation. *CALICO Journal, 21* (1), 29-40.

Byrd, P. (2001). Textbooks: Evaluation for selection and analysis for implementation. In M. Celce-Murcia (Ed.), *Teaching English as a second or foreign language* (3rd ed., pp. 415-427).Boston: Heinle.

CALICO Review [web site]. www.calico.org/CALICO_Review

Chapelle, C. (2001). *Computer applications in second language acquisition: Foundations for teaching, testing, and research*. Cambridge: Cambridge University Press.

Egbert, J., & Hanson-Smith, E. (Eds.) (1999). *CALL environments: Research, practice, and critical issues*. Alexandria, VA: Teachers of English to Speakers of Other Languages.

Fischer, R. (2004, September). *How do students use CALL reading materials, and how do we know that they do?* Paper presented at the 11th CALL Conference. Antwerp.

Garrett, M., Nimri, M., & Peterson, J. (n.d.). *Software evaluation guide*. Retrieved December 16, 2005, from http://www.owlnet.rice.edu/~ling417/guide.html

Healey, D., & Johnson, N. (Eds.) (2005). *TESOL CALL Interest Section software list*. Retrieved December 16, 2005, from http://oregonstate.edu/dept/eli/softlist

Healey, D., & Johnson, N. (1997/1998). A place to start in selecting software. *Computer Assisted English Language Learning Journal, 8* (1). Also available at http://oregonstate.edu/~healeyd/cj_software_selection.html

Hot Potatoes [authoring software]. Available at http://hotpot.uvic.ca

Hubbard, P. (1988). An integrated framework for CALL courseware evaluation. *CALICO Journal*, 6 (2), 51-72. Also available online at http://calico.org/journalarticles.html

Hubbard, P. (1996). Elements of CALL methodology: Development, evaluation, and implementation. In M. Pennington (Ed.), *The power of CALL* (pp. 15-33). Bolsover, TX: Athelstan.

Hubbard, P. (2001, March). *Extending & enhancing interactional sequences in tutorial CALL*. Paper presented at the annual CALICO Symposium, Davis, CA: March, 2001. Available at http://www.stanford.edu/~efs/phil/Hubbard-CALICO01.mht

Hubbard, P. (2004). Learner training for effective use of CALL. In S. Fotos & C. Browne (Eds.), *New perspectives on CALL for second language classrooms* (pp. 45-68). Mahwah, NJ: Lawrence Erlbaum.

Hubbard, P., & Bradin Siskin, C. (2004). Another look at tutorial CALL. *ReCALL, 16* (2), 448-461.

ISTE software evaluation form. Available at http://cnets.iste.org/teachers/pdf/App_D_Software.pdf

Jamieson, J., Chapelle, C., & Preiss, S. (2004). Putting principles into practice. *ReCALL 16* (2), 396-415.

Jamieson, J., Chapelle, C., & Preiss, S. (2005). CALL Evaluation by developers, a teacher, and students. *CALICO Journal, 23* (1), 93-138.

Kolaitis, M., Mahoney, M., Pomann, H., & Hubbard, P. (in press). Training ourselves to train our students for CALL. In P. Hubbard & M. Levy (Eds.), *Teacher Education in CALL*. Amsterdam: John Benjamins.

Levy, Michael (1997). *Computer-assisted language learning: Context and conceptualization*. Oxford: Clarendon/Oxford University Press.

Levy, M., & Stockwell, G. (in press). *CALL dimensions: Options and issues in computer assisted language learning*. Mahwah, NJ: Lawrence Erlbaum.

Longman English Interactive [computer software]. (2003). New York: Pearson.

National Foreign Language Resource Center (Hawaii). Multimedia software evaluation checklist. Available at http://www.nflrc.hawaii.edu/NetWorks/NW31/evaluation_checklists.htm

National Foreign Language Resource Center (Hawaii). [Software database]. Available at http://www.nflrc.hawaii.edu/NetWorks/NW31/software_eval

Phillips, M. (1985). Logical possibilities and classroom scenarios for the development of CALL. In C. Brumfit, M. Phillips, & P. Skehan (Eds.), *Computers in English language teaching* (pp. 25-46). New York: Pergamon.

Randall's Cyber Listening Lab [web site]. www.esl-lab.com

Reeder, K., Heift, T., Roche, J., Tabyanian, S., Schlickau, S., & Gölz, P. Toward a theory of E/Valuation for second language media. In S. Fotos & C. Browne (Eds.), *New perspectives on CALL for second language classrooms* (pp. 255-278). Mahwah, NJ: Lawrence Erlbaum.

Richards, J., & Rodgers, T. (1982). Method: Approach, design, procedure. *TESOL Quarterly, 16* (2), 153-68.

Robb, T., & Susser, B. (2000). The life and death of software. *CALICO Journal, 18* (1), 41-52.

Rosetta Stone [computer software]. available at http://www2.rosettastone.com/en/?a=b

Susser, B. (2001). A defense of checklists for software evaluation. *ReCALL, 13* (2), 261-276

Tell Me More [computer software]. Available at http://www.auralog.com/us/tellmemore6_GB.html

Underwood, J. (1984). *Linguistics, computers, and the language teacher: A communicative approach*. Rowley, MA: Newbury House.

APPENDIX

CALICO Journal Review Form (http://calico.org/CALICO_Review/softrev00.htm)

Software Evaluation Outline: Courseware
(2500 words ± 500)
3-4 Screen Shots
Name of product
Reviewer

Product at a glance (DO IN POINT FORM SUMMARY)

Product type	(e.g., drill & practice, tutorial, game, simulation, concordancer, facilitate tool, assessment, instructional management, authoring, etc.)
Language(s)	
Level	(Beginner, intermediate, advanced; child, adolescent, adult)
Activities	(e.g., multiple choice, fill-in exercises; pronunciation, dialog repetition; listening comprehension; transcription; vocabulary learning, data base building, etc.)
Media Format	(Floppy Disk, CD-ROM, DVD, WWW)
Operating System(s)	
Windows	(Version)
Mac	(Version)
Hardware requirements	
PC	(CPU)
Mac	(CPU)
RAM	
Hard Disk Space	
CD-ROM	(x speed)
DVD	
Sound	(e.g., Sound card, microphone)
Video	(x colors; screen resolution)
Supplementary Software	(e.g., QuickTime Ver x; HyperCard Ver x; WWW browser Ver x, Plugins)
Documentation	(e.g., user's Guide, Teacher's Guide)
On-line	
Printed	
Price:	
Single User	
Multiple Copies	
Site License	

(DO REMAINDER OF REVIEW DISCURSIVELY)

1. General description (25% of review)
Summary of features
Background information
Advertising claims, previous reviews
Documentation: On-line help, printed manuals

2. Evaluation (60% of review)
Technological Features
Simplicity of installation (adequacy of instructions, trouble-free, easy to un-install)
Speed of program operation (where are the delays: at startup; loading videos; web page loading?)
Reliability of operation (crashes & stalls)
Platform compatibility (PC/Mac; OS/Browser versions)
Screen management (esthetics, navigational transparency)
User interface (ease of use, operational consistency, on-line help)
Exploitation of computer potential (**effective** use of sound, graphics, video, speech recognition, speech synthesis, intelligent response handling, student record keeping, adaptability based on user profiles, www connectivity).

Activities (Procedure)
This is essentially a matter of determining what students **do** when they use a program and how well these activities are designed. Judgments here must always be made relative to activity type. You may be personally opposed, for example, to the use of structuralist grammar exercises; but in evaluating these you can't criticize them for not being collaborative in nature. You have to judge such exercises relative to how well done they are as a structuralist activity. (The appropriateness of activities is a separate issue, dealt with under Teacher Fit). Broadly speaking, activities can be classified into three major types:
Instructional (tutorials, drills, text reconstruction)
Collaborative (games, simulations, discussion forums, peer group writing)
Facilitative (dictionary, database, verb conjugator, spell/grammar checker, authoring system)
Obvious activity features to consider are:
Linguistic focus (discourse, syntax, lexis, morphology, spelling, pronunciation)
Language skills (reading, listening, writing, speaking)
Sociolinguistic focus (information gathering/authentic tasks)
Supplementary/Complementary/Central relationship to the to curriculum

Teacher Fit (Approach)
An assessment of teacher fit primarily involves looking at the theoretical underpinnings of student activities, judging how well they conform to accepted theories of cognitive development, second language acquisition, and classroom methodol-

ogy. Linguistic accuracy (i.e., grammaticality, authenticity, typos etc.) and the appropriateness of socio-cultural representations (e.g. stereotypes, gender bias) also contribute to how well a program meets teacher expectations.

Teacher fit is the most critical parameter of software evaluation, for it determines the pedagogical soundness and appropriateness of the program. No matter how technically brilliant a program may be or how rich the activities it provides, if its methodology is dubious, if it fails to adhere to its avowed instructional approach, or if it pays insufficient attention to linguistic accuracy or sociocultural authenticity, then it will be of limited usefulness.

Not surprisingly, the assessment of teacher fit is the most difficult software parameter to determine. Partly, this is because software developers do not always explicitly state the theoretical/methodological assumptions underlying their program, thereby obliging a reviewer to extract them by implication. On the other side of the coin, software producers are very much aware of what methodological approaches are in favor (e.g. communicative, learner-centered, constructivist, experiential) and label their products accordingly whatever the truth of the matter may be.

Learner Fit (Design)
In considering learner fit, you are in essence defining the intended user of the software program. In doing so, you are also determining the extent to which the program is appropriate for, or can be adapted to, the needs of particular kinds of students. Properties affecting learner fit include:
Linguistic level (grammar, vocabulary, register)
Response handling (error correction, feedback)
Adaptation to individual learner differences (age, interests)
Learning styles (recognition, recall, comprehension, experiential learning)
Learning strategies
  Field-dependent /-independent learning
  Deductive/Inductive reasoning
  Visual-graphic/Visual-textual learning
Individual/Group work
Learner control (sequencing, content, operating parameters)
Design flexibility/modifiability by the instructor

3. Summary (4-5 sentences + rating chart)
Scaled rating (1 low-5 high)
Implementation possibilities:
Pedagogical features (relative to evaluation parameters):
Sociolinguistic accuracy (typos, grammatical errors, stereotypes):
Use of computer capabilities (multimedia whistles & bells):
Ease of use (student/teacher):
Overall evaluation:
Value for money:

4. Producer Details
Developer/distributor
Name
Address
Phone
Fax
E-mail
WWW

5. Reviewer information
Biodata (75 words)
Contact addresses (phone , fax, e-mail, s-mail)

About the Author

Dr. Philip Hubbard directs the English for Foreign Students Program in the Stanford University Language Center. He has published on different aspects of CALL such as CALL methodology and evaluation of courseware and has also designed software applications for ESL.