LINGUISTICS DEPARTMENT - STANFORD UNIVERSITY

An Invitation to CALL

Foundations of Computer-Assisted Language Learning

Click here for PDF

An Invitation to CALL

Unit 2: Courseware Evaluation, Development, and Implementation

OVERVIEW

This unit looks at the sub-field of tutorial CALL from the perspectives of both the end users (teachers and students) and developers. It introduces the term *courseware*, which refers to software that is used to support formal language learning. In practice, *courseware* has been used to refer to everything from complete software packages that can be used without a teacher to software that is just a part of a language learning course, sometimes a minor or optional supplementary part. We will use the term interchangeably with that of *tutorial software* to include any software designed for language learning purposes. The objective of this unit is to give you a peek at the three dimensions of tutorial CALL--developing courseware, evaluating courseware, and implementing courseware in your classes. Although CALL courseware has arguably lost its dominant position in CALL over the past decade, it is still widely used and continues to be a significant part of the field. At the very least, it is worth exploring so that you can make an informed decision about whether to incorporate it in your own teaching or recommend it to your students for independent study. It is worth noting that more and more courseware, much of it free, is showing up on the web rather than in CD-ROMs, and that there is non-CALL courseware that can sometimes be adapted for language learning purposes.

As a backdrop to this, in a series of papers from 1987 to 1996, I attempted to develop a comprehensive methodological framework for CALL that integrated development, evaluation and implementation. The CALL world has turned out to be more complex than that original vision (it did not anticipate the rise of CMC (Unit 3), for example and other uses centered on the computer as a tool), but it still serves a purpose in laying out areas of consideration for any software that has an identifiable teaching presence. The framework expanded on an earlier one

by Martin Phillips (1985) and used the Richards and Rodgers (1982) framework (Method: approach, design, and procedure) as an organizing scheme to characterize the apparent relationships between elements of language teaching and learning and the computer. The driving force behind it was the observation that existing approaches to instructional design and in particular evaluation did not pay sufficient attention to language learning or else limited themselves to specific teaching approaches. I will be introducing a simplified version of the framework here.

ORGANIZING PRINCIPLES

Development, evaluation, and implementation are part of a logical progression in any situation that has an end product. If a company produces a computer program for balancing your checkbook, for instance, they need to 1) design it with the needs of the end users in mind, 2) evaluate it in house and encourage outsiders to review it, and 3) have a mechanism to implement it, including figuring out how to make it available and training end users in its effective operation. Of course this can be and often is cyclic rather than linear, with the feedback from evaluation and implementation providing data for subsequent development.

CALL software is a bit different from a checkbook balancing program in that it involves a more complex view of who the evaluators and end users are. Evaluation, for instance, may be connected to the developer and be used for improving the courseware prior to release, or it may be done by an outside reviewer for a professional journal. It may also be done by an individual teacher representing a school or institute, selecting materials for his or her own class, or even blogging for the wider language teaching community. It may even be by a student evaluating for possible use or purchase, or to communicate impressions to other users. As Chapelle (2001): http://llt.msu.edu/vol6num1/review1 notes, evaluation can be done judgmentally at the level of initial selection, based on how well-suited a piece of software *appears* to be, and it can also be done empirically, based on data collected from actual student use.

Development, evaluation, and implementation are thus simultaneously part of a logical progression of a courseware project and interacting manifestations of its reality. This is true whether the project is for CALL or for some other educational purpose. However, the specific domain of language teaching and learning imposes on these three a set of considerations that are not exactly the same as we would find in courseware for, say, history or chemistry or math. The framework that follows addresses those considerations. This is a revised and simplified form of the content in Hubbard (1996) and in the papers listed below (see references). The others go into more depth in language teaching approaches (1987), evaluation (1988), and development (1992). Note that an updated version for evaluation can be found in Hubbard (2006): www.stanford.edu/~efs/calleval.pdf, also covering Chapelle's (2001) framework and evaluation checklists.

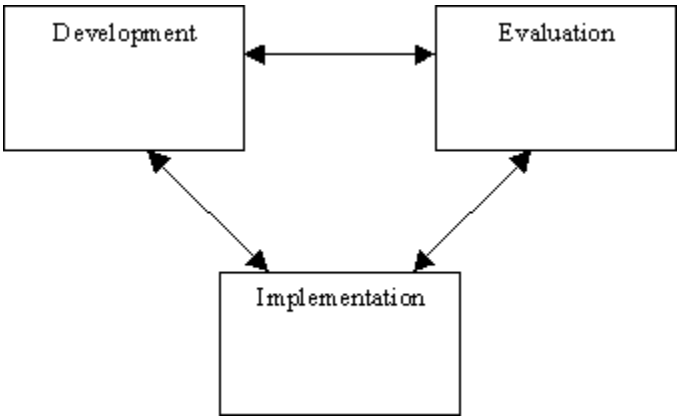
Two final notes. First, in an extensive critique of this framework in Levy (1997) http://llt.msu.edu/vol2num1/review/levy.html argues that "Hubbard's framework for CALL materials development, which assumes that all CALL is tutorial in nature, is not generally applicable to the computer as a tool. Similarly, the Richards and Rodgers model...only has limited application for the computer as tool" (p. 211). I think there may be more applicability

than he suggests, but for now we will follow Levy's view and assume this is a framework for **tutorial** CALL only.

Second, like Richards and Rodgers' framework, but unlike most others for CALL, there is an attempt to be agnostic here with respect to what actually constitutes good language teaching through computers. For the field as a whole, we need a framework which can be used equally by those whose language teaching approaches might be as diverse as those of grammar-translation, lexical, communicative, sociocultural, or interactionist proponents. Thus the framework is *descriptive* rather than *prescriptive.*

FRAMEWORK FUNDAMENTALS

The three modules (development, evaluation, implementation) share core components inspired by Richards & Rodgers (1982). In each case their original components are adapted, interpreted, and supplemented to include the reality of the computer as the interface between the teacher/developer and materials and the learner. (Realistically, in any tutorial program there IS a teacher (or at least a teaching presence) in addition to the materials themselves.) The development and evaluation modules are most closely related in terms of the elements considered. Implementation feeds on the output of evaluation. However, each module can impact the others over time, as when information from evaluation and implementation is returned to developers for updates, patches, or considerations in later versions of the product.

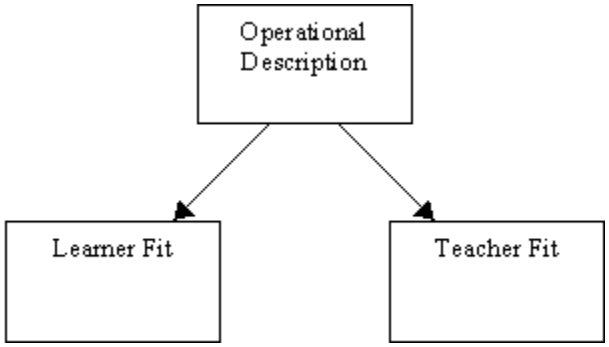


CALL Framework Interrelationships

EVALUATION MODULE

Evaluation involves three kinds of considerations. A crucial aspect is to understand what the courseware does first before attempting to judge it: this is, not surprisingly, difficult to do because as soon as we start interacting with a program we *want* to judge it. If an evaluator wants to approach the problem a little more objectively, the first consideration then is the *operational description* of the software, which essentially focuses on the procedure level elements. The design elements essentially can be subsumed under the label "learner fit." That is, based on the information from the operational description, you are looking to see how well the design elements (see Development Module, below) of language difficulty, program difficulty, program

content, etc. fit the students you are evaluating for. The approach elements, in this case approach-based evaluation criteria, can be subsumed under the label "teacher fit"--broadly, what does the software appear to represent in terms of assumptions about what language is and how language is learned, and how compatible are such assumptions with those of the teacher doing the evaluation? More generally, what kind of "teaching" is the software likely to be doing? Ultimately, then, evaluation consists of getting a clear understanding of what the software actually has in the way of material and interaction, and then judging how closely it fits with the learner's needs as determined by their profiles and learning objectives (perhaps themselves determined by a course syllabus) and your own language teaching approach. This relationship is sketched below.



CALL Evaluation Framework

It is worth noting that a modified version of this framework is still used by the CALICO Journal, www.calico.org/ for its courseware reviews. See the resources and references sections below for more details about this and alternative conceptions.

ALTERNATIVES

The methodological approach I present here has proved useful over the years but there are at least two others that deserve mention, especially once we begin to look beyond tutorial CALL. First, despite some of the limitations and biases in checklists, they have persisted over the years. Another general approach, that of building a framework on theoretical principles derived from SLA research, is seen in the work of Chapelle (2001): http://llt.msu.edu/vol6num1/review1. She identifies six general evaluation criteria, usable not only for software but more broadly for CALL tasks: language learning potential, meaning focus, learner fit, authenticity, impact, and practicality. As noted above, these criteria are relevant for both judgmental purposes and for evaluating outcomes. An example application of the framework can be seen at https://www.calico.org/memberBrowse.php?action=article&id=133.

DEVELOPMENT MODULE

Courseware development refers to the process of going from the idea of creating a piece of tutorial software through the final product. It should be informed by general principles of instructional design. However, I believe it is also critically important to recognize the

pedagogical aspects specific to language learning that traditional instructional design approaches may overlook.

In describing the development module, I review the key terms which are part of the evaluation module as well. This captures the intuitive realization that the deliberations important in deciding whether or not to use a piece of software are the same as the deliberations taken in producing the software in the first place. Like the development module, both the evaluation and implementation modules rely on versions of Richards and Rodgers' categories of *approach*, *design*, and *procedure*. However, the framework diverges from Richards and Rodgers in attempting to account for individual language programs rather than whole methods.

Approach. A language teaching approach is taken to include assumptions about what language is and how languages are learned. These are manifested in software by considering them in light of the realities of the computer delivery system. The result is a set of approach-based software design criteria.

Design. The design portion of the development framework draws heavily from Phillips (1985) and includes elements such as the following: language difficulty, program difficulty, program content, language focus (also called program focus), skill focus (also called learner focus), and learner styles supported. These interact with considerations such as the learner profiles among the intended users (age, first language, etc.) and the objectives of the syllabus and should maintain consistency with the approach-based design criteria.

Procedure. The procedure considerations are much more specific to the computing environment. They begin with the concept of "activity type", which is the general form of the exercise, e.g. text plus comprehension questions, picture identification, text reconstruction. The presentation scheme determines how the language will be presented, including the modalities (text, audio, graphics, and video) and what the general interaction sequence will be. Elements related to the presentation scheme are the screen layout, help and control options, the form of input judging, and the feedback in response to the input.

Courseware Package. The preceding elements are the considerations to be made in creating the software itself. The overall courseware package may also include an accompanying textbook, tutorial, documentation, record keeping, or other utilities (e.g., for authoring additional material).

IMPLEMENTATION MODULE

Implementation considerations are relevant during the evaluation process, but they become crucial when deciding how best to use software that is available. Some of the key questions to address in implementation are the following.

- What is the setting in which the students will be using the software (classroom, lab, home, etc.?)
- What kinds of training or preparatory activities are warranted?
- What kinds of follow-up activities either in or out of class will there be?

- Given the options provided by the program, how much control will the teacher exert, and how much control will be left to the learner?

Whether they are done in class together, in a lab with individuals or pair working on computers, or outside of class at a computer cluster, the student's own computer, or even on a mobile device like a cell phone, computer exercises should be clearly linked to the rest of the course. This does not mean they have to be fully integrated. Arguably, activities with CALL courseware can be supplementary or complementary to the classroom part of the course (including the virtual classroom in an online setting), required or optional, and still be useful. However, the instructor needs to be sure that learners see the connections and that the computer work is compatible in terms of content, level, and approach to the rest of the course material and activities. For a more detailed description of the components to consider in implementation and their interrelationships, see Hubbard (1996).

RESOURCES FOR EVALUATION

Besides the evaluation framework presented here, it is common to see evaluation checklists or other procedures. Here are a few examples.

CALICO's Online Software Review Collection (https://www.calico.org/p-21-Software%20Reviews.html)
Guide for Using Software (http://www.cal.org/caela/esl_resources/digests/SwareQA.html) in the Adult ESL Classroom by Susan Gaer
A Place to Start in Selecting Software (http://www.deborahhealey.com/cj_software_selection.html) by Deborah Healey & Norm Johnson
Bibliography of CALL Evaluation (www.nflrc.hawaii.edu/aboutus/ithompson/flmedia/evaluation/callbib.htm) and Evaluation Features for CALL Multimedia (www.nflrc.hawaii.edu/Networks/NW31/evaluation_checklists.htm) from the National Foreign Language Resource Center at the University of Hawaii
TESOL CALL Interest Section Software List (http://www.eltexpert.com/softlist/index.html): perhaps the largest list available

RESOURCES FOR DEVELOPMENT

Additional instructional design issues. As noted, there is a need to consider CALL software development, as well as evaluation, and implementation, as it relates to the general field of instructional design. In making that consideration, however, it is important to be aware that instructional design often takes a "training for mastery" approach that is not appropriate for many aspects of language learning. A recommended text for developers is Clark & Mayer (2003).

Advice for courseware design along with examples of CALL activity types for courseware can be found on the ICT4LT site site mentioned in Unit 1: http://www.ict4lt.org/en/en_mod3-2

Some authoring options:
Hot Potatoes (http://web.uvic.ca/hrd/halfbaked/): a web-authoring tool for interactive exercises--an excellent place to start for non-programmers.
HyperStudio (www.mackiev.com/hyperstudio/index.html): an authoring package built along the lines of Apple's defunct Hyper-Card)
Revolution: (www.runrev.com/home/product-family/): See also http://www.edvista.com/claire/rev.
HTML and web-page authoring tools (like Dreamweaver, www.adobe.com/products/dreamweaver); Google Sites http://sites.google.com.
Adobe Flash (www.adobe.com/products/flash/), Director (www.adobe.com/products/director), and Captivate (www.adobe.com/products/captivate).

SUGGESTED ACTIVITY. Visit the CALICO Review website at https://calico.org/p-21-Software%20Reviews.html. Find an interesting-looking piece of software and read the review, noting 1) what you can learn from it and 2) any questions that arise that might help inform your own evaluation process. If you feel energetic, try two or three. You should note the difference between a published review intended for a wide audience and your own evaluation, which should be situated with respect to your own approach, your students' abilities and needs, and the environment of your class. Also, take a look at the demos on the Hot Potatoes site (http://web.uvic.ca/hrd/halfbaked/) to get an idea of how traditional multiple choice, matching and fill-in exercises can be easily made for use on local computers or the web.

REFERENCES

- Chapelle, Carol (2001). *Computer Applications in Second Language Acquisition*. Cambridge: Cambridge University Press.
    - Clark, R.C., Mayer, R.E. 2003. *e-Learning and the Science of Instruction*. San Francisco: John Wiley & Sons.
    - Hubbard, Philip (1987) "Language Teaching Approaches, the Evaluation of CALL Software, and Design Implications" in Smith, W.F. (ed.) *Modern Media in Foreign Language Education: Theory and Implementation*. Lincolnwood IL: National Textbook.
- Hubbard, Philip (1988) "An Integrated Framework for CALL Courseware Evaluation" *CALICO Journal* 6.2: 51-74.
- Hubbard, Philip (1992) "A Methodological Framework for CALL Courseware Development" in Pennington, Martha and Stevens, Vance (eds.) *Computers in Applied Linguistics: An International Perspective*. Clevedon UK: Multilingual Matters.
- Hubbard, Philip (1996) "Elements of CALL Methodology: Development, Evaluation, and Implementation" in Pennington, Martha (ed.) *The Power of CALL*. Houston: Athelstan.
- Hubbard, Philip (2006). "Evaluating CALL Software," in Lara Ducate and Nike Arnold (eds.) *Calling on CALL: From Theory and Research to New Directions in Foreign Language Teaching.* San Marcos, TX: CALICO.
- Levy, Michael (1997). *Computer-Assisted Language Learning: Context and Conceptualization*. New York: Oxford.

- Phillips, Martin (1985). "Logical Possibilities and Classroom Scenarios for the Development of CALL," in Christopher Brumfit, Martin Phillips, and Peter Skehan (eds.) *Computers in English Language Teaching: A View for the Classroom.* Oxford: Pergamon.
- Richards, Jack and Rodgers, Ted (1982). Method: Approach, design, and procedure. *TESOL Quarterly 16.2.*

Last modified: January 21, 2010 by Phil Hubbard