Features or tasks?

Heilesen, Simon

Publication date:
2005

Features or tasks?

Simon B. Heilesen

Dept. of Communication, Journalism and Computer Science, Roskilde University
simonhei@ruc.dk

Like many other universities Roskilde University, Denmark, is preparing for a future with greatly increased use of information and communication technology in all educational activities (Cheesman et al. 2002). One challenge among many is to choose E-learning software that is not only efficient but also suitable for local conditions. In our case we particularly need software that will make it possible to remediate on to the internet the type of problem-oriented project group work, which is the hallmark of Roskilde University pedagogy. Experiments with various CSCL and CSCW software (Computer-supported Collaborative Learning/Work) have been carried out for a couple of years at Roskilde (Cheesman & Heilesen 2001, Heilesen & al. 2002, Heilesen & Cheesman 2002), but most likely the perfect system may yet have to be found.

In reviewing a number of E-learning systems currently in use at Danish Universities we have been struck by the fact that *features* (functionalities) loom so very large. Usually they define the structure of the interface; and technology rather than uses tends to dominate characterizations of E-learning systems (Heilesen & Ørum 2002). Features are of course important, but so is intended and actual use – ranging from course structure to the wider social context and underlying educational philosophy.

Below, I shall argue that in developing innovative E-learning systems, especially if constructivist pedagogy is to be applied, it will be useful to model the user interface on the often complex tasks that the user has to perform. It is an old story, really: Instead of expecting students to adapt to the way that the E-learning system works, the E-learning system should be adaptable to the ways in which humans naturally prefer to work.

Old-fashioned E-learning

There are a couple of hundred products on the market that have been designed for – or at least to some extent can be used for – Computer-supported Collaborative Learning. Even if I have tested only a fraction of them, it seems fairly safe to say that most of them offer a rather narrow range of features, and that generally they do so in fairly unimaginative ways.

Let us consider a somewhat simplified generic product in terms of possibilities for interaction:

Student/program interaction will consist mainly in download of files of course content contained in folders named fx. "course documents", "assignments", "library" etc. The objective is to provide the student with materials for self-study.

Student/teacher interaction is taken care of by a mechanism for handing in assignments and reports and getting feedback, either from an actual instructor or from a grading tool. There may also be an e-mail feature allowing for asynchronous one-to-one communication between student and teacher. The objective is evaluation and possibly also tutoring.

Student/student/(teacher) interaction unfolds in asynchronous threaded discussions forums and possibly also in synchronous chat. Some systems provide for limited-access group workspaces where students can collaborate on a project. The objective in all cases is to reach a reflected understanding by means of discussion.

Historically, the types of student/program and student/teacher interaction described above derive from the correspondence course, while the student/student/(teacher) interaction is an electronic remediation of the classroom discussion. The tools thus being really quite conventional, whatever innovation E-learning has to offer – if any – must originate in the use made of the tools.

Software reviews tend to emphasize features – the more features the better. But normally they do not go into detail about how these features are to be used or even combined. Thus they deal with what we might call the primary level of design in E-learning systems, which is *the functionality and user interface provided by the software manufacturer*, but not with the secondary level which is *the design of the educational module to be taught using the software package*.

The purpose of any good software package being to make life easier for the user, the basic design level ought to provide a satisfactory framework for most typical tasks, so that the secondary level of design may be reduced to providing content by filling in the necessary slots. Such a practice minimizes the instructor's or course designer's workload and the need for technical qualifications. If you just follow the implicit pedagogy built into the program you will end up with perhaps well-trodden but quite safe teaching patterns such as (using the example described above of a generic system):

Reading assignment → written assignment → grading and feedback

Reading assignment → moderated discussion → summing up.

This kind of instructor-directed predominantly instructivist teaching works very well in many contexts, particularly if the purpose is mastery of skills. The architecture – keeping functions separate – makes sense if the functions are to be combined sequentially for a series of operations evolving over time. The scheme is tidy from the point of view of programming and system maintenance. Also it does not strictly impose any particular work form (as do in fact some systems claiming to be suitable for constructivistic learning), and it may even be argued that it helps the user keep his bearings once he has adapted to the system.

A different need

But the fragmentation of activities may present a challenge if you wish to apply a different learning objective and pedagogical approach. Let us consider a case where you need to perform some complex tasks involving several technical features in a short time span. An example could be a constructivist learning environment where students are working on a project individually or in a group. Whether problem-based or problem-oriented it is a fairly popular teaching method in the Nordic countries, and it is one where the work-process itself and not just the outcome is considered important. Although it is linear, as projects must be, the work process includes loops and iterations and thus is rather more complex than the teaching patterns described above.

In the project-work case the central objective is likely to be the creation of a document (e.g. a paper, a web site or just a conclusion to an inquiry process). Discussions about content, adding and commenting on references, providing reviews of literature, planning and coordinating activities are some examples of necessary supplementary activities. Performing the central task thus requires using a combination of tools and having a good grasp of all the details created by means of them.

Even if many standard e-learning systems can be used for problem-oriented project work more or less effectively, it may be better simply to use a CSCW-system. Such systems usually make no implicit or explicit pedagogical assumptions (but perhaps some on the organization of work), and their tools tend to be general-purpose. Thus, in order to create a course the instructor or course planner has to work mainly on the secondary level of design, providing an architecture, outlining activities and suggesting work patterns. It is a time-consuming process and one that also requires some technical and pedagogical expertise. The price of flexibility also has to be paid by the students, who as they engage in project work also become designers, organizing and reorganizing the workspace so as to support the task at hand. Even though a valuable learning experience, it involves a considerable overhead to the learning experience.

In terms of HCI-design these processes are interesting in that they provide us with experimental data on how people manage to collaborate in learning in cyberspace: How they choose, use and combine the various tools focusing on tasks rather than system features, how they develop work routines and how they organize the virtual classroom. Systematizing such information may help us improve methods for secondary level design of E-learning systems. Eventually such methods may be formalized so that they can be incorporated in the primary level design of E-learning systems that have been designed for more complexity in learning environment and methods than is common in most of to-day's systems.

References

Cheesman, Robin & Simon B. Heilesen (2001): "Using CSCW for problem-oriented teaching and learning in a net environment". In: Pierre Dillenbourg, Anneke Eurelings & Kai Hakkarainen (eds.): *European Perspectives on Computer-Supported Collaborative Learning. Proceedings of the first European Conference on Computer-Supported Collaborative Learning*. Maastricht. P. 708-709. Full text version: <http://www.ruc.dk/~simonhei/docs/papers/cscl2001.pdf > (2002.09.02)

Cheesman, Robin, Simon B. Heilesen, Jens Josephsen & Agnieszka Kosminska Kristensen (2002): "Scenarier i computer-medieret og netbaseret undervisning". CNCL Occasional Papers, 1.1/2002. <http://www.cncl.ruc.dk/pub/OP-1_1.pdf> (2002.02.09)

Heilesen, Simon B & Robin Cheesman (2002): " Using FLE2 (Future Learning Environment 2) in problem-oriented learning". CNCL Working Paper 3/2002. http://www.cncl.ruc.dk/pub/WP-03.pdf (2002.09.10)

Heilesen, Simon B., Mia Cudrio Thomsen & Robin Cheesman: "Distributed CSCL/T in a Groupware Environment". In Gerry Stahl (ed.): *Computer Support for Collaborative*

Learning: Foundations for a CSCL Community. Proceedings of CSCL 2002. Hillsdale, New Jersey. P. 642-643. Full text version: <http://newmedia.colorado.edu/cscl/166.html> (2002.09.02)

Heilesen, Simon B. & Henning Ørum (2002): ”CSCL/W software – det vanskelige valg”. CNCL Occasional Papers, 1.3/2002. <http://www.cncl.ruc.dk/pub/OP-1_3.pdf > (2002.02.09)