Learning Environments and Usability: Appropriateness and Complementarity of Evaluation Methods

Angelique Dimitracopoulou

University of the Aegean, Department of Preschool Education,
Learning Technology and Educational Engineering Laborator
1, Av. Demokratias, GR-85100, Rhodes, Greece
e-mail: adimitr@rhodes.aegean.gr

Abstract. Technolog -based learning environments are a specific kind of software. In this paper we initiall briefly examine some often overlooked factors that are related to the usabilit y of educational software. Then, evaluation methods are examined in relation to the overall design-development cycle. It is argued that most of the methods are complementary and may be used in different phases of a specific lear n-ing environment development. It is also argued that some methods are crucial to use for formative evaluation. Finall some research approaches, appropriate for interface design and evaluation of innovative environments are discussed.

1 Introduction

The main purpose of technology -based educational environments is users *learning*. These environments do not just offer tools facilitating task execution, as in the case of man y other software environments. Educational software besides permitting the students to execute their learning-related tasks in an easy way, they should also allow learning du ing such activities and support this learning process.

The *user interface* of most educational software is the most important part of the system, given that the user interface affects considerably eventual learning. While the design of educational software is inspired by scientific knowledge or theoretical frameworks related to learning, the transfer of such theoretical concepts to software design specifications is not a straight forward exercise. It entails a lot more than just rep esenting in a compute program the original concepts or learning theory that motivated for instance the development of an innovative learning environment. A unique feature of any design problem is that there is not a single 'optimal solution" as there are always many alternatives . *Design* is an interactive activity, involving multiple actors. Strong interaction occurs b etwee theory and practice, between constraints and trade-offs, between designers and their materials, between the design team and the users or learners. Design requires a balancing act between factors that often work against each other.

As a consequence, different environments create different opportunities for learning. A given software is distinguished from another one because its interface is made of different tools, of different entities or because theses entities act in a different way. The way the software allows the user to inhabit the "world" of the interface determines the perception

and the experiences that this user will have in that world. In other words, the user exercises a role in the world of the interface and this role is associated with certain expe ience.
Evaluation is, in general, a systematic effort to gather and interpret in a systematic way, information with which one can estimate the worth or value of simple design decisions or even of a whole learning and educational enterprise.
A range of educational software evaluation methods has been defined, that can focus on the interface and point out its usability. To take advantage of the 'wonderful di versity' of these methods and techniques that are increasingly prominent in learning environments, we must seek from evaluation studies the opportun ities they can provide us *to learn*: t learn how to design better systems and how users or learners can interact in powerful and significant ways.
Usability is a self-evident requirement for all kinds of software. Given that the purpose of an educational software is not just to perform a task, but to promote learning, it may appear the following paradox: there are often cases where a seamless fluency of use is not conducive to deep learning, but merely restrain it (Mayes & Fowler 1999).

This paper discuss and points out issues related to a set of basic questions concerning educational software addressed to young and older students of primary and secondary education:

- What are the specificities of technolog -based learning environments and in particular of *educational software*?
- In what extend, the design of these systems takes into account the *context of real school use*?
- What are the general *usability evaluation* approaches? Are they exclusive or may complement each other?
- What are some explicit or even implicit criteria for selecting an evaluation method and what are the current tendencies?

2. Educational softwa e and their specificities in usability requirement

In order to examine educational software usability issues, it is useful to survey the different kinds of educational software. There are many different categories, developed during at least the last two decades. The most distinct categories currently used in primary and secondary educ ation, are presented in Figure 1.
Simulation systems are systems that simulate the behaviour of a phenomenon, of an app aratus or a machine. The user can handle variables that influence a given phenomenon and examine possible alternative representations of data. The significance in interface design lies on the facility to create new situations (in the case of open simulation systems), on the appropriateness of variables to handle and on multiple represent ations that are needed in order to visualise the data of the simulated phenomenon.
Modelling systems allow expressing and exploring models in one or multiple modelling formalisms and symbolic languages. Additionally to the simul ation systems they must allow learners to express models in an appropriate way for them.
Media Based Laboratory systems incorporate software that captures the data from real experiments (e.g. in chemistry or physics) and visualise these data by different epresenta-

tions and diagrams that constitute the main crucial aspect of the corresponding software interface design.

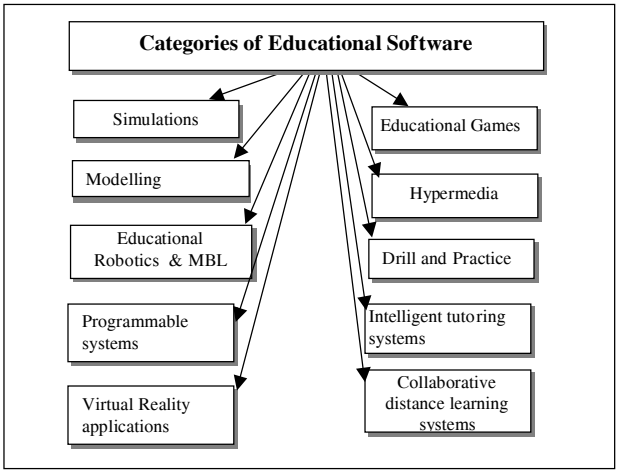

Figure 1.: Main categories of educational software

Programmable systems, are systems that allow students to program with a set of basic entities and, in general, to see the execution of their program in a graphical space, (as it happens in LOGO or LOGO-like environments). The main interface design decisions are related to the basic programming entities, visualisation of program execution a nd the functionality of the tools allowing extension and reusing of the basic entities (co nstructs).
Educational robotics systems consist usually of a possible set of external devices, also accompanied by specific software permitting to program these devi ces and eventuall visualise their state as well as their evolution.
Virtual Reality (VR) applications, may be included in various others systems (e.g. simulation or modelling systems). Their specificities are due to the 3D graphics producing VR desktop or VR embedded systems (with external specific devices). An appropriate interface design is actually crucial, given that this category of educational software still emains in its infancy.
Educational games addressed to young children or to students constitu te an extended category that incorporates various approaches in interface design as well as in educational value.
Hypermedia-based learning systems can be found in great numbers as commercial pr ducts and are produced in an explosive way after multimedia technology development of the last decade. Some of them present pieces of content formed by multimedia (text, video, animation, sound) and propose some simple questions to users. Some others ar more encyclopaedia-like, presenting with one or more ways information usually related to a specific thematic issue (e.g. the machines and how they work). Usability of these s stems is related to navigation approaches through hypermedia and to the multiple views and possible alternative structures of material. Design hypermedia interface guidelines fo adults are also applied in a great extend and in an efficient way in hypermedia systems fo students.

Drill and practices software are also widely available. They present short pieces of content and usually a set of questions (more often multiple choice or fill-in type) or simple problems. Their usability is mainly related to the feedback provided to the users, the components presenting the performance of students in various ways (global, local, concept related), the sequence of the proposed questions or exercises, etc.
Intelligent tutoring systems or intelligent problem solving systems, are mainly prototypes produced by research laboratories, and their usability lies mainly on their possibility to assure a flexible interaction when in the same time provide an elaborated feedback adapted to learners cognitive resources and needs.
Collaborative Distance learning systems are systems that may belong in different previous categories, but they have specific components and tools that allow collaboration through action and dialogue by distance, in a synchronous or asynchronous way. Their main usability related questions have to do with the action and dialogue management among collaborative partners.
All these environments present also differences related to the age of pupils on which they are addressed. The requirements for the interface design are not the same when the software is addressed in young children of pre-school education and when it is addressed to students of secondary education.

As discussed in the previous section, in the case of educational software, the purpose is not just to perform a task, but to promote learning during user activity. An easy to use system may help the student execute a task, restraining in this way *meaning making*, *understanding* and *deep learning*. For instance, let us consider a modelling system, designed in such a way that it undertakes most of the modelling activity, presenting directly to the student the appropriate model. In this case, the student will manage to resolve the task (find the model of a situation) in an efficient manner, but in this way, the software essentially prohibits efficiency in the essential purpose of the activity, which was *learning*. In other words, the educational software first of all should have as a prime objective to make the learner think.
Recent findings in education and cognitive psychology point out that learning environments should:

- Support expression of learners by *pre-existing knowledge structures* and support i the same time their eventual *evolution* during interaction period (that leads to a need of multiple expression modes or adaptable interfaces) (White & Frederisksen & 1987)
- Support *thinking and reflection*, that could imply that the software should be restricted, for instance, in providing an immediate and direct feedback (Land & Hannafin 2000), which is considered one of the main characteristics of usable software.
- Support *meta-cognition*: the development of meta-cognitive ability is an important and crucial parameter in learning process (Vosniadou 1994). This implicates that in exploratory environments (simulation, modelling, programmable systems) or in complex collabo ative ones, supplementary components and tools must be provided in order to support metacognition. This principle, may lead to inclusion of a str c-tured notebook (in order for students to write down their thoughts during activity, in an organised and easily re-accessible way) or to offer various visualisations of the 'history of interaction", that give elaborated information on the significant events.

Consequently, in the case of educational software, its usability is not related directly with the efficiency and effectiveness of the task execution, but *with effectiveness and efficiency of learning* that should occur during this activity. So, usability is merely related with the extend in which an educational software supports expression, thinking, reflection and metacognitive activity, in an efficient and effective way.

3. Usability of Educational software: some trivial cases where it is not fulfilled

We have seen in the previous section, some of specific cases of educational software and their advanced requirements derived from considerations of cognitive sciences, related mainly to the task that learners have to perform interacting with a learning environment. In this session, we will briefly examine some more trivial cases where usability factors often are not taken into account.

It is known that software usability is related not only to the tasks, but also to the nature of the users and the context of use. In the case of educational software, let us consider firstl the '*user*'. Most of educational software is produced with the intention to be used not in an individual way, but in school classrooms.

The user of educational software in school has not a single general profile. In a real school classroom, an eventual user is also the teacher of the class besides the student. Very fe educational software products recognise the teacher as potential user, in most of these cases, only providing the teacher with the possibility to arrange the educational material. Ignoring teachers' role in class, or avoiding taking them into account in an explicit way, designers of educational software often fail to provide them with specific components and tools that should help them in their complex task. For instance, the teacher tasks that need support are to pursue, know and interpret the interaction and/or cognitive paths of their students in exploratory environments, and thus to understand their conceptual or proce-dural difficulties. Appropriate tools could be designed in order to analyse students' actions and provide teachers with elaborated and structured information that could support them in significant interventions. This direction that actually concerns mainly the designers of collaborative learning systems, is needed also in other software categories, in order to allow teachers to perfo m effectively and efficiently.

Lets examine know the *context* of use of educational software, concentrating in just some of the dimensions, ignoring other ones like the social, organisational context, etc. Most educational software is considered to have a *single user* during interaction. In real school context, over the world the very large portion of classes dispose approximately half of the computers for a whole class population. This means that not one single user but merely two or in some cases three students work simultaneously on the same computer with the same software. In just a few cases educational software takes into account this situation. There are many possible ways to address this problem: a) recognizing two simultaneous users: this case is followed in some problem solving-like games, where the different single users take different roles; b) offering more than one external devices, (e.g. two mouse) as some researches have already tried to experiment with young children (Abnett et al, 2001).

The above dimensions even if usually are taken into account in commercial software for adults, are often ignored in the case of educational software.

4. Evaluation methods

There are a wide variety of methods that have been developed, permitting to evaluate the appropriateness and usability of interface (Avouris 2000; 2001). We propose to distinguish the methods used in educational software, firstly, in two major categories considering the principal purpose of evaluation:

I.) *Evaluation by designers:* includes mainly formative evaluation approaches that are undertaken with the purpose to improve the design.

II.) *Evaluation by experts, policy factors and teachers:* includes approaches that attempt to provide a global and summative evaluation of, in general, a full developed system, in order to validate it and/or support decisions on commercialisation, funding or educational use.

4.1 Evaluation by designers

In the case of formative evaluation by designers we can distinguish the following three general approaches:

a) *The clinical evaluation in the laboratory:* It is based on observations of user - software interactions. The experimental setting may consist of a single student working on the software, or of a group of two students working in common. It may include also the presence (passive or more active) of a researcher or teacher, which could intervene or not. The place of the experiment is usually a laboratory (of the design team). The clinical evaluation may be based not only on a simple observation of the interaction, but also on a detailed analysis of a range of data, such as students protocols (drafts of students notes), log files of interaction, screen captures, video-tapes of the whole sequence that may further lead to a more specific analysis of dialogues among the participants, analysis of their gestures, etc. In the case where there is the presence of the researcher, it is also possible an analysis of *thinking aloud protocols* of users. The analysis of these protocols may be focused on their reasoning, related to both user interface and the activity itself (e.g. problem solving), such as is applied in *GOMS* analysis approach. Depending on the kind of the task and/or the age of the pupils, the interventions of the researcher may be more or less direct (following a *semistructured intervention protocol*) trying not to disturb the interaction process. This observation-based sequence often finishes with *semidirected -post interaction - interview* on the students' impressions of the quality of the software, and/or with a *written typical questionnaire*. It is to be noted that written questionnaires cannot be used when users are young children. The evaluations based on interaction observations may be applied in a small or an important number of users. Usually, in the former case, researchers have the possibility to extract detailed information from deep analysis of disposed data, while in the later case, often quantitative analysis are applied. Variations of the initial standard setting, may include among other alternatives: i) Different profiles of students, ii) Different versions of the software or of just one of its tools, in order to compare and identify the more appropriate design between two or more alternatives (Suthers et all 2001).

b) *The evaluation in the field of use:* In this case the evaluation takes place in a real school context. The whole session may be videotaped, but it is often needed to be focused in one or more of the participants (the teacher, one or more groups of students).

So usually, more than one video-cameras and/or microphones are needed. Depending on the design-development phase of the software under evaluation, researchers may collect apart from the data during interaction, additional data at the end of the session, by *individual* or *panel intervie* s with groups of students and/or teachers.

c) *The long-term evaluation in th e field of use:* This approach is in general applied in order to take into account the changes that an extensive use of the same software ma ask for new specifications in the interface design. Some alternatives of this approach are *participatory design* as well as *action research* method, where the same students, in the same general context, use successive versions of the same software. In this case also the exploited data may be various (videotapes, written protocols, screen captures, interviews at the end of some sessions, etc). The analysis of data, in some cases, is based on ethnographical approaches, taking into account the whole set of intervening factors (students, teachers, broader social context, organisational conditions, etc).

4.2 Evaluation by experts and teachers

In the case of external validation by experts and teachers, we can distinguish two main methods with corresponding tools:

a) *Guidelines checklists:* Check-lists provide to experts or teachers a basis to an a-lyse the software. Many checklists h ave been produced by organisms and unions during the '80s (Heller 1991). This approach has been widely criticised, at least in its use by teachers in order to select appropriate software (Squires & Peerce, 1999).

b) *Heuristics*: Set of Heuristics is addressed either to specific experts (with co m-plementary expertise) or just only to teachers. In the case of educational sof t-ware, there is a discussion on the suitability of general software heuristics, for in-stance, such as these pointed out by Nielsen (1993) and thus attempts have been made to produce heuristics more specific to educational software (Squires & Peerce, 1999), (Kordaki & Avouris 2000).

In all these cases checklists and heuristics are methods of formative evaluation, and when they are used by the designers themselves, this happens during the latest phases of devel-opment. Other approaches are also applicable, such as *Walkthrough method* (cognitive o pluralistic), mainly by usability experts.

5. Complementarity of evaluation methods

Even if the previously presented methods have important differences between them, in a general level, we could consider that most of them and their corresponding research tools could be used iteratively in a complete design development cycle (see Figure 2).

Clinical evaluation in laboratories with single users and group of users is very important in early stages of development. Similarly, in later phases of development, experiment a-tions in the context of real use (school) can give significant information. It is to be noticed that multi -focused observation can be held in advanced phases of development cycle, given that in early phases, not controlled multiple parameters involved in the process, make data interpretation particularly complex.

During development it is important to re-use the previous methods, in order to determine the efficiency of new components, or of a new design.

Concerning the tools, *questionnaires* have little value for very young students, while *think aloud protocols* are also very difficult to be used by the same population.

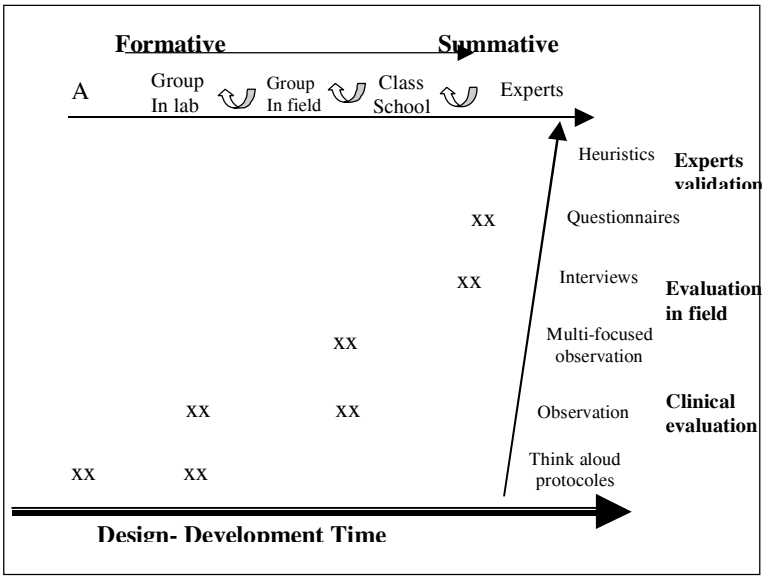

Figure 2. Design-development cycle and usability evaluation tools and methods

6. Appropriateness of evaluation methods: implicit and explicit selection criteria

Even if in a theoretical level, most of the usability evaluation approaches could be used, rare are the research groups, development teams, or even laboratories that use a wide and complete range of evaluation methods. Often research groups apply extensively specific evaluation methodologies during an intermediary period of design-development cycle and very scarcely during the 'final stage' (summative evaluation) or even a long term evaluation that could allow examine of use by experienced students.

Designers and researchers seem to have preferences in the selection of a specific methodology due to a multitude of reasons. These preferences are due to the specific category of educational software, the underlying theoretical design framework, or even may be due to the dominant academic background of researchers or simply to financial and time constraints.

The academic background of educational software research and its researchers and the discipline's relationship to other disciplines also have an effect on selection of evaluation method. Research in technolog -based learning environments may be undertaken by researchers originating from very different academic backgrounds. The most common are Computer Science, Psychology and Education Research specific to a discipline (Science Education, Mathematics Education, etc). Although related and interlinked, problems ma arise due to these disciplines belonging to different paradigms. Psychology belongs to the scientific paradigm, which lays great importance on the formal objective experiments as a

means to justify theories (although there is important groups advocating more informal techniques). By contrast, parts of computer science research are more closely associated with the engineering paradigm, which employs proof by construction; so if the software functions in the expected way, then the "theory" has been justified.
Within the engineering paradigm, one needs often to make decisions between design options, involving trade-offs on a number of dimensions. Evaluations can be used to provide information for such trade-offs between alternatives (Fidas, et al., 2001).
The frequent use of informal in-depth studies on prototypes during the development of a system is often crucial in order to reveal problems with the environment in use but also to outline general issues applicable across educational software. Researchers advocate officially the usefulness of informal-exploratory techniques (Winne 1993), against typical formal methods that fit with the scientific paradigm of objectivity and reproducibility.
Another important factor on the selection of evaluation methods is the theoretical framework (implicit or implicit) underlying the design and development process. Traditional instructional design approaches, producing simple multimedia, content based educational software are compatible with quantitative results analysis in an important number of pupils. In contrary, student-centred design approaches, influenced by recent theories of learning and cognition, such as *Activity Theory* and *Distributed Cognition* promote qualitative approaches, long term evaluations in the field of use, employing ethnographic approaches, whose importance are widely recognised.
The use of ethnographic methods in the formative evaluation of technolog -based learning environments was pioneered by Suchman (1987). Ethnographic methods are particularly concerned with the observation of behaviour in a natural setting, known as *ecological validity*. In the context of educational software, these approaches thus focus on the interactions not just between the user and the system as they occur in class, but also interactions etween the single user and the other users, the teacher(s), and other systems, computational, social, organisational, etc. Due to this, strict ethnographic analysis may be not possible to apply in formative evaluation when the prototype under investigation may not be robust enough to operate in its anticipated natural setting, particularly if that is to be the classroom. Where the intended users are not schoolchildren but adult learners, and the intended context of use is for instance self-study, or collabo ative study, in a compute lab such as the university, formative studies in a HCI or Learning Technology research laboratory may, with a little care, be undertaken without violating the principles of ethnography, and still obtain many of the benefits.

7. Conclusions

In this paper a number of perspectives relating to usability of educational software have been provided. The complex relation of usability and learnability has been investigated first. Subsequently an overview of the multifaceted world of educational software in terms of usability requirements has been outlined. Special emphasis has been provided to the specific requirements of formative and summative usability evaluation approaches applicable to educational software. The interleaving of usability and software design and development has been described. Also the need to use in a complementary way of a wide range of methods during design-development cycle in order to examine system usabilit that fulfils learning purposes has emerged from the discussion.

In conclusion, it should be stressed that the usability evaluation of an educational software, more than simple validation exercise, should provide information with significant interpretative value. Thus, it is useful to apply successive informal (i.e. exploratory) and formal evaluation methods, in laboratory as well as in real school context, in particular when the objective is to produce innovative learning environments.

References

1 Abnett C, Stanton D., Neale H. & O'Malley C. (2001). The effect of multiple devices on collaboration and gender issues. *Proc. of European Conference on CSCL*, Maastrict, The Netherlands, March 2001.

2 Avouris N., (2000) *Introduction in Human-Computer Interaction*, Diavlo Pub. Athens

3 Avouris N., (2001). Introduction to Software Usability, Workshop on Software Usability, *Proc. 8th Panhellenic Conference on Informatics,* Nicosia, November 2001.

4 Fidas C., Komis V. & Avouris N. (2001) Design of collaboration -support tools for group problem solving. In N. Avouris & N. Fakotakis (Eds), *Advances in Human - Computer Interaction*, Proceedings of Panhellenic Conference on Human-Computer Interaction PC HCI 2001, Typorama Publ., Patras, December 2001.

5 Heller R. (1991). Evaluating software: A review of the options, *Computers and Education*, 17 (4), pp. 285-291.

6 Kordaki M. & Avouris N. (2000). Evaluation methods and tools for open learning environments Tools. In V. Komis (ed). *Technologies of Information and Communication in Education*, Proceeding of 2nd Panhellenic Conference, Patras, Greece.

7 Land S.M. & Hannafin M. (2000) Student-Centered Learning Environments, D. Jonassen & S. Land (Eds) *Theoretical Foundations of Learning Environments.* LEA,1-25.

8 Mayes J.T. & Fowler C.J., (1999), Learning Technology and Usability: A framework for understanding courseware. *Interacting with Computers* 11, 485-497.

9 Nielsen J., (1993), *Usability Engineering*, Academic Press, London.

10 Squires D. & Preece J. (1999). Predicting quality in educational software: Evaluating for learning, usability and the synergy between them. *Interacting with Computers*, 11.

11 Suchman L.A. (1987). *Plans and situated actions.* Cambridge University Press.

12 Suthers D., & Hundhausen (2001). Learning by constructing collaborative representations: A empirical comparison of three alternatives. *Proc. of European Conference on CSCL*, Maastrict, The Netherlands, March 2001.

13 Vosniadou S. (1994). From cognitive theory to educational technology. In S. Vosniadou, E. De Corte, H. Mandl (Eds.), *Technology-Based Learning Environments: Psychological and Educational Foundations*, NATO ASI Serie F : Computer and Systems Sciences, Vol. 137, pp.11-18. Springer Verlag.

14 White B. & Frederiksen R. (1987). Causal model progressions as a foundation for Intelligent Learning Environments, *Report No 6686*, BBN Inc.

15 Winne P. (1993) A Landscape of Issues in Evaluating Adaptive Learning Systems. *Journal of Artificial Intelligence in Education,* V.4., N2/, Special Issue on Evaluation. pp. 309-332.