Dimitracopoulou A., (2001). Design issues in Learning Environments for young students: The importance and the limits of general usability criteria. In Avouris & Fakotakis (Eds). *Advances in Human-Computer Interaction, Proceedings of PC HCI 2001 Conference on Human-Computer Interaction,* Typorama Editions, pp. 115-120

Design issues in Learning Environments for young students: The importance and the limits of general usability criteria

Angelique Dimitracopoulou
Learning Technology and Educational Engineering Laboratory,
University of the Aegean
1, Av. Democratias, GR 85100, Rhodes, Greece
adimitr@rhodes.aegean.gr

SUMMARY
This paper initially examines some often overlooked factors (such as users and context) that are related to the usability of educational software. Then, the concept of usability is examined, trying to consider its real meaning in educational software, related more on its features supporting learning. Additionally, it is argued that user-centred or even learner-centred design approach must be actually reconsidered, taken into account recent, more social oriented, considerations. Finally, interface design and usability evaluation methods during the whole design-development lifecycle process are discussed.

KEYWORDS: Technology based learning environments, educational software, young students, human-computer interaction, interface, usability, design approaches

INTRODUCTION

The main purpose of technology-based educational environments is users learning. These environments do not just offer tools that facilitate task execution, as in the case of many other software environments. Educational software besides permitting the students to execute their learning-related tasks in an easy way, should also allow learning during such activities and support the underlying learning process.

The most important part of the educational software system is the user's interface**,** since it considerably affects eventual learning. Even when the design of the interface is inspired by scientific knowledge or theoretical frameworks that are related to learning, the transfer of such theoretical concepts to software design specifications is not a straightforward exercise. It entails a lot more than just representing in a computer program the original concepts or learning theories that, for instance, motivated the development of an innovative learning environment. A unique feature of any design problem is that there is not a single "optimal solution" since the alternatives are always many. *Design* is an interactive activity, which involves multiple actors. Strong interaction occurs between theory and practice, constraints and trade-offs, designers and their materials, designing team and users or learners. Design requires a balancing act between factors that often work against each other, making the design of educational software interface a particularly complex endeavour.

Usability, a precious and central concept of Human Computer Interaction (HCI) field, is a self-evident requirement for all kinds of software. But, given that the purpose of an educational software is not just a task's performance but mainly the promotion of learning, the following paradox may appear: there are often cases where the seamless fluency of use is not conduced to deep learning, but merely restrains it [8].
Researchers from HCI field have already slightly explored the convergencies between general usability evaluation criteria and their applicability and sufficiency in the specific category of educational software. Squires & Preece [10] as well as Kordaki & Avouris [6], point out a list of usability evaluation heuristics lying with learning quality requirements. Mayes and Fowler [8] also discuss some general design issues related to usability in three general kinds of educational software.

This paper discusses issues related to a set of basic questions concerning usability of educational software addressed to students of primary and secondary education:

- ⇨ To what extent, do the designers of technology-based learning environments and in particular of *educational software* take into account the *context of real school use, and how is usability concept mainly considered*?
- ⇨ What are the specificities of educational software and what really is the meaning of educational software's *usability?*
- ⇨ What are the current tendencies and trends in the designing process in order to assure usability?

SOME TRIVIAL CASES WHERE USABILITY IS NOT FULFILLED

It is well known that software usability is related with the performing activity and tasks during interaction, but also with the nature of the users and the context of use. According to ISO 9241-11 forthcoming standard, usability is not an intrinsic property, and it is not possible to be examined outside the particular context of use. In the case of educational software, let us firstly

consider the 'user'. Most of the educational software is produced with the intention not to be used individually by one student, but in the context of the school classroom. In spite of that, the interaction that is considered as dominant is the one that takes place between a single student and the software (see figure 1).

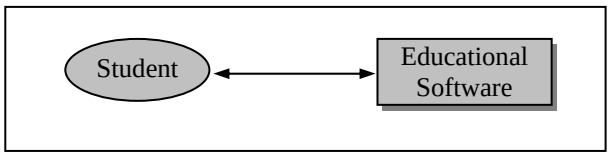

Figure 1. Dominant considered Interaction

Nevertheless, the user of educational software in the school does not have a unique general profile. In a real school classroom, an eventual user is also, besides the student, the teacher of the class. Very few educational software products recognize the teacher as a potential user. And in most of these cases, the only possibility they provide the teacher with, is the arrangement of the educational material. By ignoring the teacher's role in class, or by avoiding to take them into account in an explicit way, designers of educational software often fail to provide the teachers with the specific components and tools that should help them in their complex task. For instance, the tasks that the teacher needs support at are to be able to pursue, know and interpret the students' interaction and/or cognitive paths in exploratory environments, and thus to be able to understand their conceptual or procedural difficulties. Appropriate tools could be designed in order to analyze students' actions and provide teachers with elaborated and structured information that could support them in significant interventions. This direction that actually seems to mainly concern the designers of collaborative distance learning systems, is also essential in other software categories, in order to *allow teachers to perform effectively and efficiently*.

Let us now examine the educational software *context* of use (see Figure 2), by concentrating on just some of its dimensions and setting aside others, broader ones, like the social and organisational context. Most educational software is supposed to have a *single user* during interaction. Nevertheless, over the world, in real school contexts and in a very large proportion of school classes, the number of the computers that are being employed is approximately half the number of the whole class population. What this means is that not one single user but merely two or in some cases three (Sn in Figure 2) work simultaneously on the same computer with the same software. In just a few cases does educational software producers take this situation into account. There are many possible ways to address this problem: a) by recognising two simultaneous users: this case is being applied in some problem solving-like games, where the different single users have different roles; b) by offering more than one external devices (e.g. two mice), a case which some researchers have already tried to experiment with, by having young children [2] as their subjects.

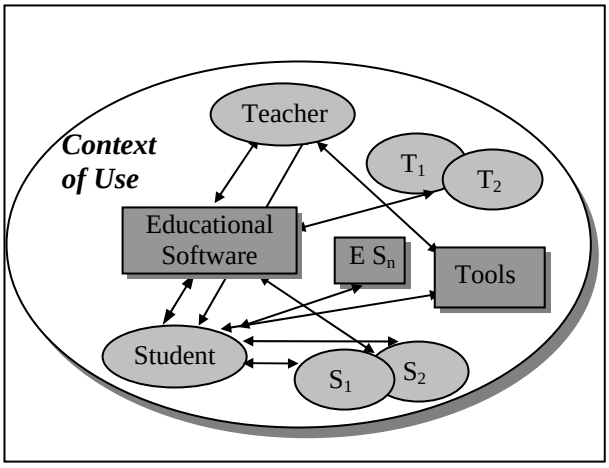

Figure 2. Interactions and Factors of the Context of use

Another factor that overlooks the context of use in real school settings is the existence of other tools and educational software ('Esn' in Figure 2) which are being used by the students in parallel. Important and unjustifiable inconsistencies with the currently used software may affect the usability, the utility as well as the learnability of a newly designed educational software.

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

EXPLORING THE MEANING OF USABILITY IN EDUCATIONAL SOFTWARE: FROM 'USABLE' TO 'MEANINGFUL'

Many different categories of educational software have been developed during at least the last twenty years. The most distinct categories currently used in primary and secondary education are**:** simulation and modelling systems, educational robotics and media based laboratory (MBL) systems, programmable systems (like LOGO, Boxer), educational games, hypermedia applications like encyclopaedias and content based software, drill and practice and intelligent tutoring systems. Additionally, during the last decade, internet based applications allowed the development of promising collaborative distance learning environments.

Most of the above systems present important differences in the interface design as well as in the requirements that relate with usability. As a consequence, different environments create different opportunities for learning. A given software can be distinguished from another, because its interface is made out of different tools and different entities or because theses entities act in a different way. The special manner by which the user of the software will be inhabited in the "world" of the interface depends on the software itself, and determines the kind of perception and the experiences that the user

will have in that world, and also the meaning that he will construct. All the environments that have been mentioned above present differences which also relate with the age that the pupils have when they are first addressed to them. The requirements for the interface design are not the same when the software is addressed to young pre-schoolers and when it is addressed to secondary education students.

Usability is defined in ISO 9241-11 as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".
Usability in general application software is related to the tasks that the user can execute with the specific software. According to this meaning, an easy to use educational software should allow the learner to perform the task with effectiveness. In this case, let us consider for instance, 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) and will have accomplished the task effectiveness. But, in this way, the software basically prohibits the *effectiveness* of the essential purpose of the activity, which was *learning* through the modelling process itself.

Similarly, in an intelligent problem solving system, the feedback of the system in the steps of the solution of the problem that the students provide, could be directly informative, by giving information on the right solution. Gradually, the software will help the student to produce the whole solution. The software helps the student to execute the task with efficiency, consuming low level resources not only in a software level but also in a cognitive level, with regard to the learner. But, this approach might also eliminate the possibilities that students have to exploit and activate their own resources, to be able to realise by themselves a reasoning approach which might be inappropriate, and, finally, find the appropriate solution on their own.

Finally, if HCI researchers or designers were addressed to the student after the interaction with these two specific educational software, it is very probable that they will ascertain the student's satisfaction from the use of the system.

The difference in the case of educational software is that the purpose is not just to perform a task, but to promote learning during user activity. And learning is a process of making meaning, not of knowledge transmission. Humans interact with other humans and with artefacts in the world and naturally and continuously attempt to make sense out of those interactions. Making meaning (resolving the dissonance between what we know for sure and what we perceive or what we believe that others know) results form puzzlement, perturbation, expectation violations, curiosity, or cognitive dissonance. This dissonance ensures that the learner has acquired a respectable amount of knowledge. In other words, an easy to use system might on one hand help the student execute a task, while on the other hand might restrain *creation of meaning, understanding* and *deep learning*.

The educational software should first of all have as its prime objective to make the learner think, reflect on his actions, reflect on the whole activity. This assumption does not set aside the activity and the task itself. There are also opposite examples of educational software, in which, for instance, in order to analyse a lot of information about cognitive abilities of students learners, they impose specific complex interactions during activity that may really frustrate any user such as the learner. We need well -balanced systems, both usable in task and not restraining in learning, assuring meaningful interactions.

The goal of user-centred software design is to make computers easier to use, thus allowing the user to focus on the use of technology in order to perform various tasks. The goal of learner-centred designers is to create software that 'makes people more effective learners' and also to design interfaces that will make them want to learn and know how to learn, beyond the computer task at hand. Recent findings in education and cognitive psychology point out that learning environments should:

- ⇨ Support the expression of learners by *pre-existing knowledge structures* and, at the same time, support their eventual *evolution* during interaction period (that leads to a need of multiple expression modes or adaptable interfaces) [14].
- ⇨ Support *thinking and reflection*, that could imply that the software should be restricted, for instance, in providing an appropriate immediate feedback [7], 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 the learning process [13]. This implicates that in exploratory environments (simulation, modelling, programmable systems) or in complex collaborative ones, supplementary components and tools must be provided in order to support metacognition. This principle, may lead to the inclusion of a structured notebook (in order for students to write down their thoughts during the 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 the effectiveness and efficiency of learning* that should occur during this activity. So, *usability* is merely related with the extent to which *an educational software supports expression, thinking, reflection and metacognitive*

activity, in an efficient and effective way; the extent in which it allows and supports learning during interaction and activity.

DESIGN PROCESS AND METHODOLOGICAL TRENDS

How can we design learning environments and how can we assure their usability? In this section we will discuss methodological trends related to the design process and usability requirement during the whole development lifecycle.

Firstly, we need to examine what differentiates an adult-user from a young student. The first important difference is that designers could not extract information that is useful on requirements directly from young learners. Learners either students or young children have not yet sufficiently developed metacognitive abilities on what they really need in order to perform tasks and achieve learning purposes. A second difference is that the goals of the interaction and the activity itself are not specified by students themselves. The objectives are usually specified by teachers, educators, or even researchers.

Given that the students themselves cannot give direct information on their needs, how can we identify goals and specify initial usability requirements?

In order to start designing, one approach could be to follow 'instructional design process' which puts emphasis on the careful, a priori engineering, of the teaching learning process, the articulation of explicit learning objectives defined by official curricula paired with systematic and empirically verifiable approaches to analysing, designing, developing implementing and evaluating instruction. This approach is often applied in conventional educational software based on content presentation and simple exercises or questions on specific parts of the content. These kinds of software, which often belong to the 'drill and practice' category, usually need simple interfaces. Instructional approaches of educational software design that develops conventional systems, have much less questions to answer during design process.

On the contrary, the innovative exploratory kind of learning environments are usually based on 'Grounded design approaches'. These are 'systematic implementations of processes and procedures that are rooted in established theory and research in human learning' [4]. Grounded approaches emphasise the deliberate alignment of core foundations and assumptions, and the linking of methods and approaches in ways that are consistent with their corresponding epistemological perspectives. According to this approach, the central design principles are derived from experimental research issues and/or from theoretical assumptions and considerations that stem mainly from the field of education (e.g. science education, mathematics education, etc.) and the field of cognitive psychology. These principles influence the initial specified requirements that may lead to the composition of a set of tools and components. The specific graphical user interface, the design of buttons, labels, and other commonly used interface elements as well as the synergy between different tools, often remains crucial. When they can not be based on existing previous experimental research in the field of learning environments design, then, they may ask for considerable design effort and experimentations with learners aiming to make comparisons among two or more alternative design approaches of specific tools interface [12].

There is a gap between requirements and design: the transformation between a represented world (requirements) and a representing world (interface). This transformation is mediated by conceptual models and metaphors, and also by a 'grammar' of the representing world – the syntax, style, and conventions of the specific implementation environment. The design of innovative learning environments is always a complex endeavour by itself, and especially in order to assure their usability.

Even if the necessity to produce usable systems is actually acknowledged for every kind of software, the remaining problem is that very often in practice, usability aspects are usually regarded very late (if at all) in software development. In educational software the situation may be more burdened.

In the case of innovative software, it is not rare to see un-usable systems in task–oriented meaning, due, for instance, to an over-demand by the system of elements during interaction in order to analyse the actions of learners and offer adapted feedback and advice, or to an over–offer of choices and tools. These cases, that often restrain system usability, lead to the frustration of learners. Approaches that could help towards this direction are participatory design and a well established experimental design agenda, studying users performing tasks.

Many experimentation approaches may be applied during development: clinical evaluations in the laboratory with single students and/or group of students, evaluations in a simulated or in a real context of use, employing research tools like observation analysis based on various data (video, sound, observation protocols, logfiles, screen interactions captures), multi-focused analysis, think-aloud protocols, interviews, questionnaires, etc., [1], [3].

It is to be noted that although experimental evaluations may be used, in the middle period of design development life-cycle, their expense will restrict them to a very few particularly significant decisions out of the very many that any design includes. The frequent use of informal in-depth studies on early and elaborated 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 [15], against typical formal methods that fit with the scientific paradigm of objectivity and reproducibility but are very

consuming in time and effort, and they are not so worthwhile in early usability evaluations.

In general, researchers seem to have preferences in the selection of a specific methodology due to a multitude of reasons [3]. 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. Traditional instructional design approaches, producing simple multimedia, content-based educational software are compatible with quantitative results analysis in an important number of pupils. On the contrary, student-centred design approaches, influenced by recent theories of learning and cognition, such as *Activity Theory* and *Distributed Cognition,* promote qualitative approaches and long term evaluations in the field of use.

Let us examine now, what happens after the basic development life-cycle. Most systems and products are modified and improved in a number of releases over a number of years. Web sites are being updated and modified continuously. However, most efforts at working with usability matters stop after the initial development process. What do we do after delivery?

In some cases, external validation approaches, accomplished by usability experts and/or teachers are applied, like the heuristic usability evaluation proposed by Nielsen [9]. In the cases of educational software has also made attempts to produce appropriate heuristics for usability evaluation [6], [10]. But this kind of method cannot but have a little impact in design and re-design lifecycle, since, by nature, it has a post-development external validation status.

Usability related work is important to be continued after first full development and during 'releases' period. Long term research, allows to examine the real context of use better, as well as to investigate usability factors for experimented users; an aspect almost completely ignored in educational software.

In those cases, the application of ethnographic experimental methods are particularly suitable. Ethnographic methods are especially 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 on the interactions between the single user and the other users, the teacher(s), and other systems, the computational, social, organisational context factors, etc.

CONCLUSIONS-DISCUSSION

The designers of educational software either they don't pay attention to usability requirements, or usually they envisage this quality factor in one-dimensional view, taking into account only the student–software interaction during task performance and ignoring complex interactions that usually occur in real school contexts.

Usability is a central concept in User-Centred design approach (UCD), in the engineering perspective. User-centred design has surfaced as the primary design approach to facilitate usable interactive systems, offering a collection of tools and methods for planning, iterative development and evaluation, while it fosters a tight evaluation feedback loop to assure that the deficiencies are identified and corrected at an early stage of the development life-cycle. The corresponding approach in the field of educational software is the so-called Learner-Centred Design (LCD). Initially, LCD approach was specified in order to be clearly differentiated from instructional design approaches where the starting point has not been the student-learner and their needs but the objectives and the content specified in the official curricula. Similarly, LCD educational software design approach is based on experimentation with learners. Even though it happens, and given that it is actually well established (at least in the research field) we argue that LCD named approach may actually lead to misunderstanding and misconceptions, reinforcing the tendencies to examine the usability in an entrenched single learner –software interaction. The real school context of educational use is multidimensional considering the factors that we have to take into account and the interactions that may occur.

Moreover, in the current states of the extended Internet exploitation and in the view of the trends in the advances and use of technology, it becomes evident that in order to provide the required support for the design of the broad range of computer supported mediated activities in the emerging virtual spaces, UCD as well as LCD approaches, as philosophies, should be extended to provide more prescriptive design framework. In the context of HCI researchers the need to refine and extend existing techniques and tools of user centred approaches with concepts from social sciences is started to be recognised [5], [11]. These researchers claim the need to provide a broader foundation for HCI design. By doing research on potential design contributions rooted at disciplines that focus on human communication in social contexts they extend the analytical design approaches with social constructs that are based on Activity theory, language action theory, situated action models, distributed cognition.

The interface design of educational software, and especially in the case of the innovative one, is an extremely complex approach. Usability engineering in learning environments is a tremendous design problem given that its real meaning goes beyond the usability on task performance and has merely to do with the extent in which a software allows and supports learning. Thus, the general usability criteria and guidelines that may be partially useful, remain limited. It should be stressed that the usability evaluation of an educational software, more than a simple validation exercise, should provide information with significant interpretative value, and should be applied not only during the whole

development lifecycle but also during the post production period. In order to examine system usability that fulfils learning purposes, a wide range of methods is often needed to be applied in a complementary way: successive informal (i.e. exploratory) and formal evaluation methods in laboratory as well as in real school context. The need of really multi-disciplinary design teams (technological, HCI, usability engineers, cognitive psychology scientists, science educators, etc) are broadly evoked, but it is needed to be also fully applied in educational software design. Additionally, usability engineering research must focus on the development of specific methods and tools of analysis that could significantly help designers to produce usable innovative educational software.

ACKNOWLEDGEMENTS

Financial support from the Network of Excellence on Software Usability, funded by the Greek Secretariat of Research and Technology is acknowledged. Special thanks are due to the members of the network for fruitful discussions.

BIBLIOGRAPHY

1. Avouris N. *Introduction in Human-Computer Interaction*, Diavlos Pub. Athens (2000).
2. Abnett C, Stanton D., Neale H. & O'Malley C. The effect of multiple devices on collaboration and gender issues. *Proc. of European Conference on CSCL*, Maastrict, The Netherlands, March 2001.
3. Dimitracopoulou A. Learning environments and Usability: Appropriateness and complementarity of evaluation methods, *Proc. 8th Panhellenic Conference on Informatics,* Nicosia, November 2001.
4. Hannafin M.J., Hannafin K.M., Land S. & Oliver K. Grounded practices in the design of learning systems. Educational Technology Research and Development, 45 (3), 101-117. 1997.
5. Hollan J., Hutchins E., & Kirsh D. Distributed Cognition: Toward a new foundation for Human-Computer Interaction Research. *ACM Transactions on Computer-Human Interaction,* Vol7, No2,2000, pp. 174-196
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. Student-Centered Learning Environments, D. Jonassen & S. Land (Eds) *Theoretical Foundations of Learning Environments.* LEA,1-25, 2000.
8. Mayes J.T. & Fowler C.J. Learning Technology and Usability: A framework for understanding courseware. *Interacting with Computers* 11, 485-497, 1999
9. Nielsen J. *Usability Engineering*, Academic Press, London., (1993).
10. Squires D. & Preece J. Predicting quality in educational software: Evaluating for learning, usability and the synergy between them. *Interacting with Computers*, 11, (1999).
11. Stefanides C, & Salvendry G. Toward an Information Society for All : HCI challenges and R&D recommendations. *International Journal of Human-Computer Interaction,* Vol.11(1), 1999, pp. 1-28.
12. Suthers D., & Hundhausen D. Learning by constructing collaborative representations: An empirical comparison of three alternatives. *Proc. of European Conference on CSCL*, Maastrict, The Netherlands, March 2001.
13. Vosniadou S. 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, (1994).
14. White B. & Frederiksen R. Causal model progressions as a foundation for Intelligent Learning Environments, *Report No 6686*, BBN Inc., (1987).
15. Winne P. 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.