Technology and Teacher Education Annual, 1996

General Editors

Bernard Robin Jerry D. Price Jerry Willis Dee Anna Willis

Associate Editors

A. C. Blackburn
Stephanie Boger-Mehall
Jenny O. Burson
Caroline M. Crawford
Michael L. Connell
Glenn L. DeVoogd
Emma Isleib
Sara McNeil
Pamlyn Reed
Kathy Rosa
Tom Troutman

Section Editors

Ray Braswell
Deborah Y. Bauder
Sue Espinoza
Carine Feyten
Marianne Handler
Nancy Hunt
Jacqueline A. Kemper
Lara Kiser
Kathryn Matthew
Dale S. Niederhauser
Ronald Sarner
Harold Strang
Neal Strudler
Keith Wetzel
Cameron White
James A. White
Nancy Williams
Susan Williams

Proceedings of *SITE 96* –

Seventh International Conference of the Society for Information Technology and Teacher Education (SITE)

Phoenix, Arizona; March 13-16, 1996

AACE Association for the Advancement of Computing in Education

Educational Software: Are We Approaching it the Right Way?

Netiva Caftori
Northeastern Illinois University

Marcin Paprzycki
University of Texas of the Permian Basin

During recent years a constant increase of usage of educational software in K-12 can be observed. To examine how such software is being used we have consulted the last three issues of the *Technology and Teacher Education Annual* (1993, 1994, 1995) *Proceedings of the Society for Information Technology and Teacher Education* (SITE; formerly Society for Technology and Teacher Education). Our assumption was that since SITE is one of the biggest conferences addressing the usage of technology in education (including teacher education), our findings will be representative of the state of the art in the area. We were definitely surprised by our findings. We have found a large number of papers addressing the usage of educational software in classroom settings.

Typical examples were based on case studies of how good results can be achieved if software is properly used to support the learning process. In most cases the educational software was used by the teacher in a class as part of a lesson (a great example of such an approach is a discussion of how the same Multimedia program -- *Henry and Mudge in Puddle Trouble* -- can be used in a variety of courses on a lower level of elementary education (Land & Taylor, 1995). The second domain that was discussed was teachers writing their own educational software. A group of papers introduced various tools that can be used by the teacher to author the educational software while another group of papers presented case studies of how teachers write software and use their own products in class. Even though we think that, at this point, the idea of teachers' writing their own educational software is highly unrealistic (primarily because of the time constraints, but also because of the software quality considerations; see below), this is not the major point we would like to address. There seems to be a big gap between the great ideas discussed during the SITE meetings and the educational reality.

In the paper of Byrum, Cashman & McCraw (1994) we found a clear indication that the technology in schools is not delivering what it has promised, generating a lot of frustration among the parents, students and educators. In the present paper we, first, present how the software is often used in educational practice, and discuss the pedagogical problems that this type of usage leads to. Second, we argue that at least one of the reasons for these problems can be traced to the inappropriateness of educational software evaluation criteria. We will show how a program can easily pass an extensive pedagogical evaluation (such as the set of criteria described by Persichitte, 1995) and still not facilitate learning. Third, we will try to suggest how the hardware and software development have led to the creation of a new set of problems that the educational software research needs to tackle. Finally we will specify the areas in which changes are necessary to overcome the problems.

Educational Software in Practice -- Unsupervised Learning

A typical example of a situation that illustrates the points raised by Byrum et. al. (1994) was presented by Caftori (1994a). She describes how, in a Junior High School, a number of computer terminals have been set up so that students can interact with the educational software and thus learn. The computers are used in an unsupervised mode. What was found was that this educational software does not (in many cases) play an educational role, or at least not the educational role it was intended to.

A very informative example illustrating the encountered problems is based on the *Oregon Trail* game (a game which proved to be very successful with children). This game is a history simulation with the educational goal of introducing children to the life of covered-wagon travelers on their way from the Missouri River to Oregon in 1848. In its educational objectives it is suggested that this game induces students to make intelligent decisions (including decisions based on a limited amount of data and considering alternative solutions when the circumstances suddenly change). It contains a number of problem-solving situa-

tions, like river crossing, money and food management and dealing with disease outbreaks. It is also supposed to teach students to arrange the data they have gathered into the "bigger picture" and to establish interrelations between the facts so that they can make appropriate decisions. The overall effect of all the decisions made determines the final outcome of the game. A number of pedagogical problems were observed (only some of them will be listed here; for a complete discussion see Caftori 1994a):

- children concentrate on reaching the end of the trail as fast as possible without regard for their companions or oxen,
- children take no time to visit the landmarks and learn their history,
- shooting animals for the sake of shooting becomes an objective in itself; the type of the terrain and the animals associated with it are not noticed and learned by the student.

A number of similar problems were observed when studying how students interacted with other games:

- using trial and error strategies (instead of calculations) in games like *Paper Plane Pilot* and *Wood Car Rally*,
- not playing *Where in the World is Carmen San Diego?* as the game takes a long time for completion (this may not be a problem in the case of at-home usage),
- playing the game of *Odell Lake* with the goal of swallowing fish and enjoying the sound effects related to it (instead of learning the predator-prey relationship).

One reason for this failure to meet the educational objectives is that the designers have forgotten that the experience has to be authentic and relevant to the learner's life (Woolf & Hall, 1995). Such relevancy could have been supplied by the teacher if the programs were incorporated in a lesson. Another reason, and probably most important (and for which there is strong agreement among specialists), is that for the time being computer-based instruction is most effective when used in conjunction with other instructional strategies (Persichitte, 1995). In an unsupervised environment (and we may include in here many computer labs of more than five computers, where it is nearly impossible for a teacher to be aware of what each student is actually doing, much less learning), there is no additional instructional strategy to supplement the software. There is no way a student can be reminded of the objectives of the lesson, and no way of bringing her/him back on track when she/he strays away.

In addition to these general problems some interesting observations have been made (in supervised as well as unsupervised environment) related to the gender differences in approaching the games (only some of the differences will be listed here; for a detailed discussion see Caftori 1994a and 1994b):

- girls were less visible in the computer laboratory and thus participated in the supplementary learning less often,
- girls in early grades prefer word games (such as *Hangman*) to construction with geometric figures (such as *Mosaic*),
- overall, boys like fast, shooting, fighting, or killing games involving battle or space ships while girls prefer slower games involving writing or school work,
- even when playing the same game (*Oregon Trail*) girls have pursued different goals (reaching the destination -- the original goal of the game, writing epitaphs on tombstones etc.),
- girls do not like software that does not allow them to quit a section in the middle; boys do not like software that does not provide them with an appropriate feedback.

In summary, even though students were interacting with the educational software, they were able to do it in such a way that at least some (if not all) of the specified educational objectives have been missed. Software attributes that were intended to attract children to the game (e.g., competitiveness) actually diverted their attention from the objectives. The observational results confirm also that much software is designed to appeal to boys without consideration of the effect it has on girls (see also Huff & Cooper, 1987). This final comment should be emphasized even more when the overall picture of gender issues in the educational software is taken into account. For instance, in the recent survey of 344 educational games only 28 featured active female leading characters (Fryer, 1995).

One may argue that the problems observed are related to the fact that most games were used in an unsupervised mode. This is definitively part of the problem. It is very easy to imagine how each of these games could be used successfully when incorporated into the lesson. As we have stated earlier, the Proceedings of the STATE/SITE meetings contain a number of examples how such a goal could be achieved. At the same time it is clear to us that unsupervised usage of educational software takes place. To explain this we must take into account a number of factors (which are also apparent when studying the STATE/SITE Proceedings).

1. It takes a long time to prepare a quality lesson which incorporates technology.
2. There is a lot of pressure to provide students with access to computers and the educational software (this pressure comes from school principals, school regional boards, state governments as well as parents).
3. Very often there is money available to public schools for purchasing computer hardware and software, but there may be no money for training in-service teachers.

In addition, we should not forget about the students' homes. The computer hardware (PC) has become a typical household appliance (similar to the TV or a cassette player). Many parents are purchasing educational software to help their children learn. We have found out that in most cases children are interacting with the computer in an unsupervised mode. Overburdened parents having limited time to spend with their children hope that they are helping them learn by investing in software that is labelled as educational. They believe that if a specialist has evaluated the software and decided that it is educational, then their children will benefit from interacting with it. (These facts have been confirmed in our conversations with students and parents).

Educational Software Evaluation

In the previous sections a number of problems occurring during the unsupervised usage of educational software were discussed. A suggestion was made that even if a part of the problem was that the educational software was used in the unsupervised mode, this problem will not be easy to eliminate. The question thus arises: how is the teacher/parent to know that the software that is being designated as educational really facilitates learning? The most natural response seemed to be: by evaluating the educational software (or by assuming that the software was properly evaluated by a specialist). It is worth mentioning here that there are at least 2000 new programs released every year, to add to the 1990 total of well over 20,000 educational programs (Geisert & Futrell, 1990). To assess the area of educational software evaluation we have searched through the STATE/SITE Proceedings. We were astonished to find only a total of *five* papers related somewhat to the issues of educational software classification and evaluation: Byrum (1993), Byrum (1994), Maddux (1993), Persichitte (1995) and Valmont (1994). This is especially intriguing when compared with the number of papers suggesting that teachers should write their own educational software. (How are they supposed to do a good job at it if they have no background in evaluating educational software and differentiating between good and bad software in the first place?) Out of the five papers Maddux discusses the (very popular) categorization of educational software into Type I and Type II software. The paper by Persichitte introduces an elaborate software taxonomy and selection strategy. Let us observe the results of applying these two approaches to the *Oregon Trail* game.

Educational Software Evaluation Using Existing Criteria

In Maddux (1993) a division between the Type I and Type II software has been proposed as a conceptual framework that should be taught to the prospective teacher. Educational software of Type II "makes new and better teaching methods available - methods that would not be possible without the use of computers" (p. 212). It is also stated that "it is Type II applications that have the potential to legitimize educational technology" (p. 213). The typical characteristics of Type II software are (pp. 213-214):

1. Type II software requires and encourages relatively active intellectual involvement.
2. Most of what occurs on the screen is determined by the user, rather than by the software developer, and the type of interaction between learner and machine usually involves a highly varied (sometimes unlimited) repertoire of acceptable responses.
3. It often requires days or weeks of use before everything the software is capable of doing has been observed.

The example of *Oregon Trail* game clearly matches all three objectives of Type II software described above: it encourages relatively active intellectual involvement; even though there is a limitation on the number of acceptable responses and despite the fact that a large part of what occurs on the screens is somehow determined by the software developer, the number of possible variations is large enough to satisfy the second characteristics; obviously the complexity of the game is large enough to pass the final criterion. It can be thus said that *Oregon Trail* is an educational software of Type II. Unfortunately this does not tell too much to the teacher about its value as educational software.

The paper by Persichitte (1995) is based primarily on earlier work, but let us examine it as she presents a very elaborate educational software taxonomy and the selection/evaluation model. Based on work by Lockard, Abrams & Many (1987), among others, she distinguishes the following types of educational software:

- drill and practice,
- tutorials,
- simulations,
- instructional games,
- problem solving.

In addition she suggests (based on Alessi and Trollip, 1991) that for effective instruction each type of educational software must have the following four basic phases present (p. 379): a) presenting information, b) guiding the student, c) practice by the student, d) assessing the student learning.

Let us see how this taxonomy could be applied to the *Oregon Trail* example. The first problem we will run into is deciding where to attach the game. As it was specified this is clearly a game, but it is also a simulation and it contains a large number of problem solving elements (at least in its educational objectives). We will assume for a while that this is an instructional game, but similar results would be obtained if either of the remaining two choices were selected. The criteria to be used to score the software are (pp. 380-381). For presenting information:

- appropriate for particular audience,
- clear, adequate directions for the learner,
- clearly directed toward an instructional goal,
- models of learner interaction vary.

It can be suggested that in each of these categories the game would score relatively high (it can be easily imagined how the educator tests the game and is able to find all the specified educational objectives — since the educator plays the game "according to the rules"). For guiding the student:

- learner understands whether the game is one of skill or chance,
- learner understands the constraints of the game,
- learner control of returning to the directions.

Here also the game would most likely score well; children know that this is a game of skill, they understand the objective of reaching Oregon (even though they often need reminding), they usually have control of returning to the directions. For practice by the student:

- collaborative decision-making encouraged,
- team competition fostered,
- color, graphics, sound controlled by the learner's decisions in the game.

Again, it is easy to imagine how the game can score well. The outcome of the game is based on decision making, so making it a team-based game is easy. As we have observed the game definitely generates competitiveness (so it could also generate team-competition). The game has an elaborate set of interactive features that are based on the decisions made by the learner. For assessing student learning:

- summative feedback message, perhaps composed by the learner.

Definitely the game scores well in this category. There is the final score given back to the learner that is based on the overall outcomes of all of the decisions made up to that point.

Persichitte suggests that each of the above mentioned categories should be scored on a scale (0-3) and the combination of all scores would lead to establishing the *Instructional Factor* rating. We hope that the reader can see how an educational game like *Oregon Trail* could reach a very high score in the *Instructional Factor* category. It can be suggested that in the remaining categories proposed by Persichitte (*Computer Appropriateness Factor, Program Technique Factor, Hardware Requirements Factor*, and possibly even the *Overall Evaluation Factor* — even though here the situation is a touch more complicated) *Oregon Trail* will also obtain very good scores. There is nothing in this taxonomy and evaluation model that can be used to address the pedagogical problems reported above.

The Origins of the Problems

We would like to suggest that many of the problems discussed above have historical reasons and reflect the rapid development of computer hardware and software.

Unsupervised Learning

In the early days of computing (before the rapid spread of PC's, computer networks and multimedia) the primary (if not the only) way for the learners to interact with a computer was in the supervised mode (at the University or in School). This impacted the development of the evaluation criteria for the educational software and possibly caused the omission of unsupervised learning.

Software Complexity

During recent years a qualitative change occurred in the area of educational software. The first generation of educational software was relatively simple drill-and-practice software or linearly sequenced software (that can be compared to reading a book). At this time word-processing, spreadsheets and database programs were categorized as Type II educational software. The new complexity of educational software can be easily observed in the *Oregon Trail* program which cuts across the currently defined types of educational software: it is game, a simulation and a problem solving program. In addition the newly developed educational software moves from a linear sequencing of interactions (including iterations and limited selections) to multilinear sequencing of interactions brought by increased processing speed, improved graphics and multimedia. Additionally the computer has gained the capability of storing information about all earlier moves and thus the ability of guiding the child better. The software complexity has increased so rapidly that it has outgrown the evaluation criteria.

Gender/Racial Issues

In the early days of computing, interacting with computers was related primarily to the computer professionals and students taking CS courses. The fact that most of the CS students at that time were white males contributed to the gender/racial stereotypes that society has developed. This may also explain why the gender/racial sensitivity issues were not included in the early educational software evaluation criteria.

Time Factor

The division of software into Type I and Type II has one very important characteristic that is not being properly recognized. Most of the early educational software was of Type I; i.e., it was simple, its utility could have been easily exhausted and it could have been easily replaced by other teaching methods -- e.g., in the case of drill-and-practice, it can be replaced by flash cards. The new software becomes very complex and its exploration can take a long time. For instance, finishing *Oregon Trail* may take hours and the

game cannot be easily replaced by another teaching strategy. How this should be taken into account in the educational software evaluation process (and how may it be related to the apparent decrease of childrens' attention span) is not immediately clear. Definitely it is a new challenge that did not exist when the original software evaluation criteria were established.

What Needs to be Done

A number of suggestions naturally follow from what we have said so far. The primary one is that the research in the area of educational software authoring and evaluation needs to be given a high priority in order to address the new issues introduced by the rapidly changing software and hardware. The primary challenges seem to be evaluation of software used in the unsupervised mode, assessment of effects of the increased complexity of software, and inclusion of gender/ racial issues into the evaluation criteria. To research the above, a number of studies similar to that done by Caftori must be performed. They must concentrate on what really happens in the unsupervised learning environment. These studies may lead to an improved communication between educators and educational software developers. For instance, it is easy to specify what types of changes could have made the *Oregon Trail* game a more valuable educational tool. (A new version of the *Oregon Trail* has been released to take advantage of the new generations of hardware. We have not tested it, so we are not sure if the pedagogical problems listed have been addressed.) For example, when a "not so wise" decision is made by the student while traveling on the trail, the software should indicate that a more efficient way may exist. When the student strays from the set goal (such as to get *all* travelers to Oregon *and* in good health) a reminder should be provided about the major objective of the game. Similar suggestions can be easily made for the remaining games discussed above. The fact that such changes do not take place shows that there is a communication breakdown between those who develop educational software and those who use it.

Finally, a number of curricular changes must take place in the areas of: general student population (where the knowledge of various aspects of computers in society should be promoted); teacher preparation (where a special course related to the usage of computers in educational settings should become a core requirement for all prospective teachers); and Computer Science education (where a human-computer interactions aspect of software development should be included). More about these proposals can be found in Paprzycki & Caftori (1995).

References

Alessi S. M. & Trollip, S. R. (1991) Computer-based instruction: Methods and development (2nd ed.). Angled Cliffs, NJ: Prentice Hall.

Bitter, G.G. & Wighton, D. (1987). The Most Important Criteria Used By the Educational Software Evaluation Consortium. The Computing Teacher, 7-9.

Byrum, D.C. (1993). Selecting computer applications: A model for pre-service teachers. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1995 (pp. 336-340). Charlotleville, VA: Association for the Advancement of Computing Education.

Byrum, D.C. (1994). Educational Software Types: An Interactive HyperCard Tutorial. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1995 (pp. 454-458). Charlotleville, VA: Association for the Advancement of Computing Education.

Byrum, D.C., Cashman, C. & McCraw, P. (1994) The Educational Computing Backlash. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1995 (pp. 107-110). Charlotleville, VA: Association for the Advancement of Computing Education.

Caftori, N. (1994a). Educational Effectiveness of Computer Software. T.H.E. Journal, 62-65.

Caftori, N. (1994b). Examination of Computer Software in Relation to Gender Differentiation. Journal of Women and Minorities in Science and Engineering, vol. 1 (3), 237-252.

Carey, D. et. al. (ed.) (1993). In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1993. Charlotleville, VA: Association for the Advancement of Computing Education.

Fryer, B. (1995, December). Sexism and Kids' Software, PC World, 406.

Geisert, P. G. & Futrell, M. K. (1990). Teachers, computers, and curriculum. Boston, MA: Allyn and Bacon.

Land, B.L. & Taylor, R. (1995). Planning and Creating Interactive, Multimedia Lessons for Literature-Based Reading Programs. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1995 (pp. 336-340). Charlotleville, VA: Association for the Advancement of Computing Education.

Lockard, J., Abrams, P. & Many, W. (1987). Microcomputers for educators. Boston, MA: Little, Brown, and Co.

Maddux, C.D. (1993). Teaching Teachers to evaluate software: A system and an example. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1993 (pp. 212-215). Charlotleville, VA: Association for the Advancement of Computing Education.

Paprzycki, M. & Caftori, N. (1995). Educational Software Literacy, submitted for publication.

Persichitte, K. (1995). Basic Criteria for Selecting and Evaluating Instructional Software. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1995 (pp. 379-382). Charlotleville, VA: Association for the Advancement of Computing Education.

Valmont, W., (1994). Electronic Children's Books: The Good, The Bad, and the Hype. In D.A. Willis, B. Robin, & J. Willis (Eds.), Technology and Teacher Education Annual, 1994 (pp. 495-497). Charlotleville, VA: Association for the Advancement of Computing Education.

Woolf, B. P., & Hall, W. (1995). Multimedia Pedagogues: Interactive Systems for Teaching and Learning. Computer, vol 28 (5), 74-80.

Netiva Caftori, Department of Computer Science, Northeastern Illinois University, Chicago, IL 60625
E-mail: uncaftor@uxa.ecn.bgu.edu

Marcin Paprzycki Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762.
E-mail: paprzycki_m@gusher.pb.utexas.edu