Software Evaluation - A Pedagogic Solution

Sivakumar Alagumalai

Jonathan Anderson

The Flinders University of South Australia

V. Mala

Ministry of Education, Singapore

Numerous methods have been advanced in educational software evaluation. They range from evaluating the technical aspects of software to examining their pedagogic strengths. This study reviews some of the advances made in educational software evaluation and highlights short comings of technical-based and content-based evaluations. It then attempts to provide a practical solution for software evaluation using sound pedagogical principles through teacher-collaboration via a networked-database. The paper illustrates this with a working example of an ‘expert-pedagogic’ database for storing pertinent details of selected sections of educational software and how it was used [is to be used] in teaching / instruction and is currently being researched and

tested at the School of Education. Details of setting-up and maintaining such a tool and implications for teachers / educators of IT, specifically with educational software are given. Teacher collaboration and the internet-based databases are also discussed.

Please send suggestions and queries to:
sivakumar.alagumalai@flinders.edu.au

Introduction

Instructional computing can be grouped into three categories: tool, tutor or tutee. These distinctions have become vague as educational software continue to evolve (Alessi & Trollip, 1991, p.3). They argue that a more general level of categorisation of computer application software and courseware would be into administration, teaching about computers and teaching with computers. However, they purport that all three modes are relevant, especially when designing educational software.

Intricately tied to these three modes is the design process of educational software. In the design of any software, the following successive stages occur: analysis and information gathering, specification or working from the required functionality to a detailed

design, implementation where the resultant system is built, and evaluation where the system if tested (Quinn, 1994, p.46). He iterates that cognitive engineering goes into the design process, and more so with educational software. Thus educational software designed for thinking is tied to software design for instruction and learning.

Although researchers have indicated that content and design are the two most important elements of good software (Diptinto & Turner, 1995; Small & Grabowski, 1992), currently available software evaluation checklist require wholistic decision making criteria (Komoski & Plotnick, 1995) which obscures outstanding sections / segments of a software. This paper illustrates some software evaluation checklists and its shortcomings. It also attempts to provide a solution for software re-use and in setting up a networked-database for teachers. Pertinent aspects of a lesson are stored in this database, and the paper attempts to show the mechanics of setting up such an information-base (infobase). Teacher collaboration and technical aspects of the infobase are discussed.

Traditional software evaluation procedures and checklists

All effective instruction requires careful planning. Teaching with instructional media like software further necessities this planning stage. To Heinich et al., (1996), teaching / learning process progresses through several stages as defined by Gagné (1968). In line

with Gagné's (1965) model, Heinich et al., (1996) have devised the ASSURE model – a procedural guide for planning and conducting instruction that incorporates media, a software as in the case here. The ASSURE model fuses the findings of software selection principles (Reiser, & Dick, 1989; McDougall & Squire, 1995; McNeil, 1996) and instructional theory (Ohlsson, 1993; Mayes, 1993; Eklun, 1995).

The ASSURE model can be summarised as having to:

A: Analyse the Learner

S: State the Instructional Objectives

S: Select Methods, Media and Materials

U: Utilise Media and Materials

R: Require Learner Participation

E: Evaluate and Revise

The above model highlight the 'generally forgotten' aspects of learning situations in software selection and evaluation, as technical aspects take preference in traditional software evaluation (McDougall & Squires, 1995). A review of currently used checklist indicate deficiencies with regards to learning outcomes (see Microsift Evaluators Guide, 1982 and Squires & McDougall's Software Evaluation Review, 1994).

Although traditional checklist do look into content accuracy, pedagogical appropriateness, and user control, to name a few, the changing design of current software, especially multimedia titles, becomes a huge undertaking for the software selectors and evaluators. Software assessment which include ‘selection, review and evaluation’ can thus be a formidable task (McDougall and Squires, 1995). However in making the final evaluation and recommendation, specific outstanding sections in a "not so popular" software is rejected and eventually forgotten. Shuell and Schuecker (1989, p.147) thus indicate that "development and evaluation of educationally sound software is still in its infancy." Hence, coupled with an inadequate assessment mechanics and an expanding resource base, one has to look beyond evaluation of software or for the matter of fact any teaching resource. It is imperative that the fate of a software title not be allowed to sit on the pen-stroke of an evaluator and definitely not on an imperfect, narrow set of criteria.

Blease (1986) in his study found that the written reports about the teaching effect of software evaluation were more consistent than the numerical scores and ratings obtained via checklists and opinionnaire. It would then be necessary to look into how suited a software is for a class, and thus view its appropriate use from the learning environment perspective.

Komoski (1987, p.84) argues that, "there has to be different sets of criteria for selection of software for different areas." This is in

line with Winship's (1988) statement that current checklists cannot allow for different teaching strategies. Reiser and Dick (1989) recommend that the best form of software assessment is through a retention-test to identify if there had been substantial learning. Reiser and Dick (1989, p.49) indicate that, "if the retention test reflects the amount of learning that may be attributed to the instruction, then it is the best indicator of the effectiveness of a software." Hence, it can be argued that the best test for any software is its 'ability' to bring about learning in a target user. This would mean even looking at a specific section of the software, and understanding its worth rather than making generalisations about it.

Practical evaluation and Infobase

As opposed to making general recommendations about a software (Komoski & Plotnick, 1996), specific sections of a software used or that has potential for classroom application can be highlighted and documented. Like bringing sections of a newspaper or a magazine article for classroom discussion and activities, parts of a software title (or softlets) can also play similar roles and can be used at various phases of instruction. Each softlet serves as an important resource and link between the teacher, student and classroom environment.

Battle and Hawkins (1996) argue that the internet is a technologically

innovative environment for teachers to explore lesson development. Complementing this online development and enhancing any teacher's lesson would be these softlets and its associated lesson plans and practical field-experience of the teacher. These valuable information provides not only the very network needed by teachers but also serves as a resource base for both practising and pre-service teachers. This paper advocates the development of such a resource for teachers – an infobase of softlets, and its associated practical processes and skills.

Lesson plans have been identified as important elements for successful teaching and learning processes (Saxe, 1992; Battle & Hawkins, 1996). These lesson plans developed and modified in real classroom situations are given further refinement when it is shared between the teaching community. Lesson plans may vary from culture to culture, but may have central connecting themes as trialled for the Science-On-line (SOL) project (Battle & Hawkins, 1996). In the SOL project, teachers were given control over the design and implementation of text content, image content and organisation and presentation of their lessons, but adhered to commonly agreed framework. The general plan adopted for the infobase incorporates structure of the SOL and the direction advocated by the ASSURE model (Heinich et al., 1996).

Although the subject, topic and level intended for are important index criteria for a lesson plan, softlet's title and related content areas are important and were included. Figure 1 summarises the key

subheadings / fields for the infobase.

Level:

Subject:

Topic:

Software Title:

Publisher (URL):

Softlet Theme:

Softlet's Location:

Instructional Phase: Pre-, Instructional, Post-Instructional

Lesson Plan: (Details with Time-Frame)

Field Observation (Remarks):

Activities (if any):

Contributor:

Email:

Addendum:

Figure 1. Infobase Template

The above template can be modified over time, especially in the case

when multimedia moves from standalone workstation type to distributed multimedia or networked multimedia (Nicolo & Sapio, 1996).

Currently numerous multimedia packages developed by teachers through HyperStudio, Toolbook and Authorware are available on the internet through shockwave (Minoli & Keinath, 1994; Sapio & Nicolo, 1995). For these softlets available on the internet, the Uniform Resource Locators (URLs) would be indicated in the Publisher's field. The implications and potential for such an infobase are far reaching and enables the global expansion of teaching / learning, multimedia, technology and knowledge itself.

Infobase and supporting cognitive theories

Instructional software can be self running and could provide the necessary learning. Sweeters (1994) indicates that the currently available multimedia titles with full motion video, sound and extensive interactive branching have become more dynamic and powerful learning tools. However, numerous researchers (Shuell and Scheckler, 1989; Tolhurst, 1992; Akpinar & Hartley, 1996) contend that not all of them teach effectively. The problem is further compounded when the aims of the softlet are not in tandem with the cognitive requirements of the learner (Ohlsson, 1993; Kulik & Kulik, 1996).

Traditionally, education software like all other application software

have been developed by system scientists and hardcore software programmers, who may have limited knowledge of educational processes involved and its operation in the classroom. In light of this problem, Shuell & Scheckler (1989, p.136) argue that educational software development should be consistent with current knowledge about teaching and learning.

The infobase proposed here leans heavily on psychological perspective that combines cognitive theories of teaching and learning, with lesson plans being practical manifestations of these theories, and softlets which have been specifically identified, selected and trialled by practising teachers.

Sweeters (1994, p.47) purports that an efficient lesson plan, even though modelling the ASSURE model, should embrace the 'functions of learning' through the 'Events of Instruction' as advanced by Gagné and Briggs (1979). Figure 2 summarises the 'Events of Instruction'.

Events of Instruction

Gain learner attention
State the objectives
Recall the prerequisities

Present information
Provide learning guidance

Elicit expected performance

Provide feedback

Assess performance

Enhance retention

Figure 2. Robert Gagné's Events of Instruction

The introduction and use of softlets fit well in Events 1, 4 and 7. Thus softlets can provide vital support at the pre-, instructional and post-instructional phases. The availability of synchronised lesson plan and softlets, set against well thought out objectives, learning tasks and the appropriate practice and assessment would bring together the potential of two powerful systems.

Each teaching / learning node, the node encompassing a lesson plan and its associated softlet(s), could focus on a specific concept or skill. A couple of nodes could build-up or be grouped in a meaningful way by topic or subject to form a system of learning hierarchical networks for effective teaching. Thus the infobase is further tested against both time and in various educational environment as teachers around the globe both contribute and become consumers of this pool of information. With the expanding amount of information available, teachers can cope with what is practically available to them, and trust that it has been trialled by a contemporary.

As educational software evolves into a powerful force, the infobase provides the teacher these synergies within their grasps. The infobase brings together the advantage of an electronic learning / teaching systems (Sweeter, 1994), the power of electronically shared lesson plans (Battle & Hawkins, 1996) and at the same time coping with the demands of our learners (Shuell & Schueckler, 1989). The advancement of networking of professional teaching elements through the infobase further supports the notion of both evolving cyberspace pedagogies and methodologies advanced by Anderson and Alagumalai (1996).

Setting up the Teacher's Infobase

Common-Gateway-Interfaces (CGI) and Integrated Development Environment (IDE)-links are the backbone engines for the Teacher's Infobase. The HTTP protocol used by the internet is generally a one-way street, going from servers to clients. However, browsers can ask the server to display specific requests. Thus there is also the return requester path. CGIs function on this path (McComb, 1996). CGIs pass data in two ways: one is URL-based and can be displayed readily while the other is hidden. Commonly used CGI functions are GET and POST.

McComb (1996, p.550) indicates that "CGI programs that use the GET method are generally easier to write, but the URL is limited to 256 characters. The POST method is ideal when lots of data has to be

provided by the client, and there are no restriction to the number of character used." An example of these CGI functions is attached in Appendix A.

Tied to these CGI functions are the necessary IDE-links that passes commands from the CGIs to the databases that stores the information. The Microsoft's Access *.mdb database format was used as it provided several advantages over other database engines (Garcia, 1997). The front-end panel was a form designed using JavaScript and HyperText Markup Language (HTML) (see Figure 3).

3. Introductory page of Infobase including menu (right)

This HTML form provide the client with the 'search infobase, infobase presentation, and submit data to infobase' facilities. Figure 4 summarises the general structure of the infobase.

C

L

I HTML (Form)

E

N

T

CGI (Server)

S

E

R

V

E

R .EXE

C Database (*.mdb)

L

I

E

N

T HTML (Form/Panel)

Figure 4. Technical structure of Infobase

The Access database had fields corresponding to the subheadings of the infobase (Appendix B) and sat in a directory where the CGI functions were located. Website Professional Version (Beta) II was used to launch the infobase on the internet. The server software supports both the database engine and the compiled CGI functions.

Microsoft's Index Server (MIS) was used as the expert search engine. The search mechanics of the MIS supports the basic query language. At its simplest, a query can be just a word or phrase of a field in the infobase. Figure 5 is a sample of the infobase search engine.

Figure 5. Infobase search engine

The MIS also allows for searches of combinations of various key fields using the operators AND, OR and NOT. This facility allows for refinement of searches and displays the required information on a HTML panel (Appendix C).

Implications of Teacher's Infobase and Conclusion

The infobase apart from providing the necessary information for teachers, facilitates use and re-use of software, which may otherwise have been rejected due to the wholistic software evaluation process and inadequate criteria. Furthermore, a dynamic, ever growing and continuously refined practical teacher information is developed to cope with information and technological explosion. Teachers need not be passive recipients of information but active contributors shaping methodology for the newer generation of learners.

In line with the argument advanced by Anderson and Alagumalai (1996), there is an inevitable shift in both pedagogies and methodologies. With institutions heading towards flexible delivery, it is timely that useful teacher's supporting information be made available. The infobase is a humble direction proposed and trials of full implementation have been successful. The success lies not in conquering technological capabilities, but in winning over the interests of teachers.

Apart from providing the best form of resource for teachers, the infobase would also be an important asset for researchers, especially for those doing comparative work in curriculum and information relating to teaching and learning. Cross cultural and between country comparisons would be facilitated by this dynamic infobase. In line with softlets being the main focus of this database, it would mean shifting the paradigm from teacher as software-user to that of software-designer. The infobase would thus put teachers in command of the direction of future educational software development. Indeed the infobase is a force to be reckoned with.

References

Alessi, S.M., & Trollip, S.R. (1991). Computer-based Instruction: Methods and Development. Englewood Cliff, NJ: Prentice Hall.

Anderson, J., & Alagumalai, S. (1996). From text-based pedagogy to cyber-based methodology. Paper presented at the 10th Joint Conference of the Australian Association for Research in Education and the Singapore Educational Research Association, Singapore (25-29 November 1996).

Battle, R., & Hawkins, I. (1996). The study of emerging teacher practices in internet-based lesson plan development. Journal of Science Education and Technology, 5(4), pp.321-342.

Blease, D. (1986). Evaluating Educational Software. London: Croom Helm.

Diptino, V., & Turner, S. (1995). Zapping the hypermedia zoo – Assessing students' hypermedia projects. The Computing Teacher, 22(7), pp. 8-11.

Eklund, J. (1995). Cognitive models for structuring hypermedia and implications for learning from the world-wide-web. [Online]. Available: _ HYPERLINK

http://aysweb95/papers/hypertext/eklund/index.html

__http://aysweb95/papers/hypertext/eklund/index.html_, (19/3/96)

Gagné, R.M., & Briggs, L.J. (1979). Principles of instructional design. New York: Holt, Rinehart and Winston.

Garcia, J. (1997). Make your database sing. VisualBasic Programmer's Journal, 7(9), p.95-97.

Heinich, R., Molenda, M., Russell, J.D., & Smaldino, S.E. (1996). Instructional Media and Technologies for Learning. Englewood Cliffs, NJ: Prentice Hall.

Komoski, P.K. (1987). Educational Microcomputer Software Evaluation. Eurit86. Oxford: Pergamon Press.

Komoski, P.K., & Plotnick, E. (1995). Seven Steps to Responsible Software Selection. ERIC Digest Document. New York: Office of Educational Research and Improvement.

Kulik, C.C., & Kulik, J.A. (1991). Effectiveness of computer-based instruction: an updated analysis. Computers in Human Behaviour, 7, pp. 75-94.

Mayes, J.T. (1993). Commentary: Impact of cognitive theory on the practice of courseware authoring. Journal of computer Assisted

Learning, 9, pp. 222-228.

McComb, G. (1996). JavaScript Sourcebook. New York: John Wiley & Sons, Inc.

McDougall, A., & Squires, D. (1995). A critical examination of the checklist approach in software selection. Journal of Educational Computing Research, 12(3), pp. 263-274.

McNeil, S. (1996). A practitioner validated list of competencies needed for courseware authoring. Technology and Teacher Educational Annual, Association for the Advancement of Computing in Education, pp. 564-570.

Minoli, D., & Keinath, R. (1994). Distributed multimedia through broadband communications. Norwood, MA: Artech House.

Nicolo, E., & Sapio, B. (1996). Structural Analysis of gobal multimedia scenario: Technological, Market, Environmental, and Regulatory Issues. Journal of Instruction Delivery Systems, 10(1), pp.17-27.

Ohlsson, S. (1993). Impact of cognitive theory on the practice of courseware authoring. Journal of Computer Assisted Learning, 9, pp. 194-221.

Peled, Z., Peled, E., & Alexander, G. (1992). A Taxonomy for computer software adoption policy. Journal of Educational Computing Research,

8(1), pp.81-100.

Quinn, C.N. (1994). Designing educational games. In Beattie, K., McNaught, C., & Wills, S. (Ed.) Interactive Multimedia in University Education: Designing for Change in Teaching and Learning. Amsterdam, The Netherlands: Elsevier Science B.V.

Reiser, R.A., & Dick, W. (1989). Evaluating instructional software. Educational technology Research and Development, 38(3), 43-50.

Sapio, B., & Nicolo, E. (1995). Analysing the management context for global multimedia projects. In Proceedings of Internet Symposium. St Petersburg (Russia). Sept. 1995, pp. 163-171.

Saxe, G. (1992). Studying children's learning in context: problems and prospects. The Journal of the Learning Sciences, 2(2), pp.215-234.

Shuell, T.J., Schueckler, L.M. (1989). Toward evaluating software according to principles of learning and teaching. Journal of Educational Computing Research, 5(2), pp.135-149.

Small, R.V., & Grabowski, B.L. (1992). An exploratory study of information-seeking behaviours and learning with hypermedia information systems. Journal of Educational Multimedia and Hypermedia, 1(4), pp. 445-464.

Squires, D., & McDougall, A. (1994). Choosing and using educational software: A Teacher's Guide. London: Falmer

Sweeters, W. (1994). Multimedia electronic tools for learning. Educational Technology, 34(5), pp. 47-52.