Evaluating Computer Programs: Tools and Assessment

Wajeeh Daher

Abstract:

One goal of this paper is to review efforts that attempt to provide tools for evaluating computer programs in general and educational computer programs in particular. These tools include open tools and closed tools in the form of 'forms' which in turn include an online form which gives the score for the computer program which you evaluate, depending on your own evaluation. Another goal is to review researches that assess how teachers and preservice teachers assess computer programs. This would serve teachers and preservice teachers to select evaluation tools that fit their own needs and benefit from past experiences of teachers and preservice teachers who evaluated computer programs.

***General tools*:**

Software attributes:

Buckleitner (1999) suggests considering the following aspects when evaluating software: (1) What is software? (2) What is the intended purpose of the software, and where is the software intended to be used? (3) What is the developmental level of the intended audience? (4) How does the software compare with similarly designed, competitive products? (5) What theoretical orientation do you bring to the software evaluation process? (6) Does the software take advantage of the latest technology? (7) What is the history of the software in question, and what is the current "state of the art" of comparable software?

Wheeler (2005) suggests *a* general approach to evaluate computer programs, especially Open Source Software, where 'open source software' programs are 'programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program, and to

redistribute copies of either the original or modified program (without having to pay royalties to previous developers)'. This general approach is based on four steps: identifying candidates, reading existing reviews, comparing the leading programs' attributes to one's needs, and then performing an in-depth analysis of the top candidates. Important attributes to consider include functionality, cost, market share, support, maintenance, reliability, performance, scaleability, usability, security, flexibility, and legal/license issues. Below I describe each one of the criteria and aspects depending on Wheeler (ibid).

***Identifying candidates*:**

Wheeler suggests that, in order to identify programs' candidates for the program that you need, you should 'ask friends and co-workers, particularly if they also need or have used such a program', not only to ask but 'If they have experience with it, ask for their critique; this will be useful as an input for the next step'. Wheeler also suggests searching the internet for the program that you need and suggests some search engines like 'google' and gives tips on how to do the search.

Reading existing reviews:

Wheeler suggests that after identifying the options and reading existing evaluations about the alternatives, it's time for learning about a program's strengths and weaknesses from a few reviews, which would be better than trying to discern that information just from project websites. Here too, Wheeler gives suggestions how to search for the reviews and how to be careful regarding biased reviews.

Comparing the leading programs' attributes to one's needs:

The goal of comparing the leading programs' attributes is to shorten the list of realistic alternatives to a few "most likely" candidates' and suggests to perform the comparison after reading a few reviews, because 'the reviews may have identified some important attributes you might have forgotten or not realized were important'. Wheeler notes that often we can quickly eliminate all but a few candidates.

Performing an in-depth analysis of the top candidates:

Wheeler suggests that the attributes to be considered, in the in-depth analysis of the top candidates should include: functionality, cost, market share, support, maintenance, reliability, performance, scaleability, usability, security, flexibility, and legal/license issues. These attributes are described below in more detail.

- ***Functionality*:**

  Does the program do what you want it to do?

  Issues that should be considered when considering 'functionality': 'how well it integrates and is compatible with existing components you have', 'If there are relevant standards (de jure or de facto), does the program support them?', 'If you exchange data with others using them, how well does it do so?', and 'will the hardware, operating systems, and related programs it requires be acceptable to you?'.

- ***Cost:***

  When considering costs, we should consider all costs related to deploying a program, which is done by computing the total cost of ownership (all costs related to deploying the program over a period of time) or as a return for investment (by comparing the total costs to the total benefits), over a fixed period of time. Wheeler emphasizes that the costs that we consider should include for each option all costs, 'such as initial license fees, installation costs, training costs, support/maintenance costs, license upgrade fees, transition costs (such as data transition and/or transitions to upgrades), and the costs of any necessary hardware'.

- ***Market share:***

  This aspect is concerned of 'how popular a computer program is'.

- ***Support:***

  'The term "support" covers several areas: training users on how to use the product, installing the product, and answering users who have specific problems trying to use a working product (including suggesting work-arounds for weaknesses in the current product).

- ***Maintenance/Longevity:***

  When examining the maintenance we should first examine the developer mailing list archives for evidence that the developers are discussing improvements to the software, whether there are multiple developers (so that if one is lost, the project will easily continue) and whether the developers regularly check regarding improvements and bug fixes. We also examine here if their version management

information is accessible to the public and in general whether there is evidence that the software is under continuous development.

- ***Reliability:***

  Wheeler remarks that reliability is difficult to measure and depends on how the program is used. It should be noted that problem reports are not necessarily a sign of poor reliability. The best way to measure reliability is to try it on a "real" work load.

- ***Performance:***

  The best way to measure performance is to try the computer program on a "real" work load specific to our circumstance.

- ***Scaleability:***

  Wheeler describes scaleability as the size of data or problem that the program can handle. Examining scaleability means finding some evidence that the program has been used the way we want or expect.

- ***Useability:***

  Useability concerns measuring ‘the quality of the human-machine interface for its intended user’. A highly useable program is a program which is easier to learn and easier to use.

- ***Security:***

  Evaluating a product's security is related to the specific environment that the user requests and different environments impose different security requirements on the same product.

- ***Flexibility:***

  Flexibility measures how well a program can be used to handle unusual circumstances that the program wasn't originally designed for.

Examining flexibility means looking if there are mechanisms that make the program adaptable for new purposes.

- ***Legal/license issues:***

  Legal issues are primarily defined by a program's license. Examining license issue means examining the license requirements for each program that we consider, as well as their implications in the country where we want to use the computer'.

Quality factors:

Martin and Shafer (1996) suggest seven 'quality factors' which could serve to measure the four 'quality areas': maintainability, evolvability, portability and descriptiveness. The factors are: consistency; independence; modularity, documentation, self-descriptiveness, anomaly control and design simplicity. The relationships between these seven quality factors and the four quality areas are shown in figure (1) below:

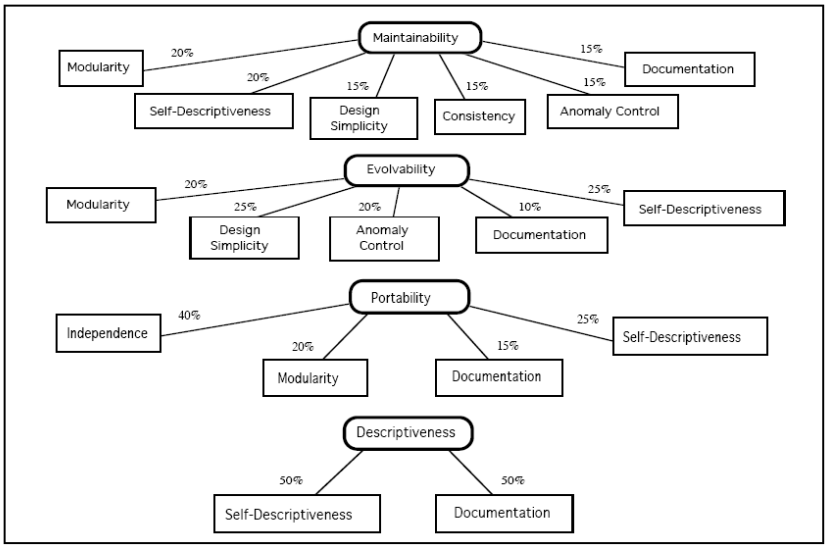

Figure 1: Quality Areas to Quality Factors Map

Note that the four ‘quality areas’ are defined by their components and the percentages of these components described in figure (1).

Martin and Shafer summarize representative questions showing the seven factors as follows:

- Consistency: Have the project products (code and documentation) been built with a uniform style to a documented standard?
- Independence: Have ties to specific systems, extensions, etc. been minimized to facilitate eventual migration, evolution, and/or enhancement of the project?
- Modularity: Has the code been structured into manageable segments which minimize gross coupling and simplify understanding?
- Documentation: Is the hard copy documentation adequate to support maintenance, porting, enhancement and re-engineering of the project?
- Self-Descriptiveness: Does the embedded documentation, naming conventions, etc. provide sufficient and succinct insight into the functioning of the code itself?
- Anomaly Control: Have provisions for comprehensive error handling and exception processing been detailed and applied?
- Design Simplicity: Does the code lend itself to legibility and traceability where dynamic behavior can be readily predicted from static analysis?

Software product evaluation standard:

(From the International Organization for Standardization)

Dobrica and Niemela (2002) describe the software product evaluation standard from the International Organization for Standardization. This international standard defines six characteristics that describe, with minimal overlap, software quality.

- Functionality

- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Below is the description of every characteristic as stated in Dobrica and Niemela (2002):

***Functionality*:**

Functionality is the set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

Reliability:

Reliability is the set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Usability:

Usability is the set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

Efficiency:

Efficiency is the set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Maintainability:

Maintainability is the set of attributes that bear on the effort needed to make specified modifications.

Portability:

Portability is the set of attributes that bear on the ability of software to be transferred from one environment to another.

***<u>Educational evaluation tools</u>*:**

The three tools that we described for evaluating computer software have some components in common and vary in other components, but they are all tools for evaluating general computer programs. Educators were interested in tools for evaluating educational computer programs. We describe some of the tools developed for the evaluation of educational programs.

Fernandez (1997) talks about four main areas to address when evaluating educational programs:

- Content - subject matter, aims and objectives, bias, concepts introduced, relevance, flexibility and teaching style.
- Usage - preparation required by the students and teachers, prerequisite knowledge, operation in the lecture theatre/laboratory, technical details, follow-up activities.
- Features - operation of the software, input of data, presentation of material, program structure, level of customization available.
- Support materials - technical and user manuals, availability and quality of teacher and student materials (lesson plans, work sheets, etc.), packaging.

We see that each of the four main areas that Fernandez describes is involved with the educational side of the computer program, for example, the 'Features' area, is concerned, among other things, with the presentation of the study material.

Gerdt, Miraftabi and Tukiainen (2002) present an outline of a checklist, the TUP-model, that covers the technical, usability, and pedagogical aspects of the educational environment so that more complete evaluations may be created. Each of the three aspects includes several different issues. The *technical aspect* includes the availability, maintainability and ease of initial setup and administration of the environment. The *usability aspect* includes the learnability, efficiency, and memorability of the user interface. Gerdt, Miraftabi and Tukiainen add to the three components the issue of perceptual and motor factors: "in addition to the three focus areas we need to evaluate how well the environment takes into account perceptual and motor factors as well as the information factors related to the environment's use" and talk about the overlap area between the technical and usability aspects: "An overlap between the technical and usability aspects can be found when the visual aesthetics and internationalization abilities of the environment are taken into account". The third component; *the pedagogic context* includes the aspects: supported educational approaches, pedagogic soundness of the content, supported types of the interaction, the possible integration of the evaluated software environment with other educational materials. Gerdt, Miraftabi and Tukiainen remark that 'Focus on how the environment motivates its users (e.g. internal vs. external motivation) and the environment's support for self-evaluation chart more learner-related characteristics is needed'.

Stamelos et. al (2000) describe a twofold process for evaluating educational software: the educational aspect and the technical aspect. The technical aspect includes the same six features that we saw before in 'Software product evaluation standard from the International Organization for Standardization'. Stamelos et. al (ibid) describe *thoroughly* each of the

features. It should be noted that here each of the features is divided further into other features that make the functions of the original features more obvious than in the other attributes and features that I described above. The features with their sub-features are described below.

***Functionality*:**

Functionality is defined as the degree of existence of a set of functions that satisfy stated or implied needs and their properties. In the case of educational software, these functions and properties may concern the coverage of one or more required subjects, the presence of experiments, various types of exercises, etc. Functionality includes the following aspects:

- Suitability which is the degree of presence of a set of functions for specified tasks.
- Accuracy which is the degree of provision of right or agreed results or effects.
- Interoperability which is the degree to which the software is able to interact with the specified systems (i.e. physical devices).
- Compliance which is the degree to which the software adheres to application-related standards, conventions or regulations in laws and similar prescriptions.
- Security which is the degree to which the software is able to prevent unauthorized access, whether accidental or deliberative, to programs and data (i.e. login functions, encryption of personal data, etc.).

Reliability:

(Exactly as previously defined by the International Organization for Standardization)

Reliability includes the following aspects:

- Maturity which is the frequency of failure by faults in the software. In general, any fault due to software problems is unacceptable for educational software.
- Fault tolerance which is the ability to maintain a specified level of performance in cases of software faults or infringement of its specified interface.
- Recoverability which is the capability of software to reestablish its level of performance and recover the data directly affected in case of failure.

Usability:

Usability is defined as the effort needed for the use by a stated or implied set of users. This attribute affects also the educational effectiveness of a software product, since if the product is hard to use, the attention of the trainee is mostly focused in the software itself, than in its educational content. Usability includes the following aspects:

- Understandability which is the user's effort for recognizing the underlying concept of the software. This effort could be decreased by the existence of demonstrations.
- Learnability which is the user's effort for learning how to use the software.
- Operability which is the user's effort for operation and operation control (e.g. mouse support, shortcuts, etc.).

Efficiency:

(Exactly as previously defined by the International Organization for Standardization)

Efficiency includes the following aspects:

- Time behavior which is the software's response and processing times and 'throughput' rates in performing its function.
- Resource utilization which is the amount of resources and the duration of such use in performing the software's functions.

Maintainability:

(Exactly as previously defined by the International Organization for Standardization)

Maintainability includes the following aspects:

- Analyzability which is the effort needed for diagnosis of inefficiencies or cause of failure or for identification of parts to be modified.
- Changeability which is the effort needed for modification, fault removal or for environmental change.
- Stability is the risk of unexpected effects of modifications.
- Testability is the effort needed for validating the modified software.

Portability:

(Exactly as previously defined by the International Organization for Standardization)

Portability includes the following aspects:

- Adaptability which is the software's opportunity for adaptation to different environments.
- Installability which is the effort needed to install the software in a specified environment.
- Conformance which is the degree to which the software adheres to standards or conventions related to portability.
- Replaceability which is the opportunity and effort of using the software in the place of specified older software.

Evaluating the educational effectiveness:

Stamelos et al (2000) state that, in contrast to the technical aspect, there is no broadly accepted model for assessing the educational effectiveness of the software, for the following reasons:

- It's very hard to describe the context of all possible educational software evaluation problems with a single attribute framework. The process of evaluation carried out by a teacher or a student would be different from the process of evaluation carried out by an educational institution.
- There are several types of educational software products, for example 'drill and practice', 'tutorials', simulations', instructional games, and 'problem solving'. Stamelos et al say that each of these software types may need different evaluation attributes or aspects.
- An educational software product may have such original characteristics that prevent the use of a predefined set of evaluation attributes.

Stamelos et al proposed a set of educational attributes for evaluating the education aspect of a software product, not before they remarked that this set of attributes must be viewed as a general evaluation framework that in most cases should be adapted to the specific circumstances of an evaluation problem. The educational aspect that Stamelos et al propose has two parts which are by themselves decomposed into other aspects. Table (1) shows this construct as in Stamelos et at (2000).

• **Educational features**
- Target users' specification. - Information for topics addressed and learning objectives. - Instructional support materials. - Adaptation to individual needs. - Strategies for enhancing engagement, attention and memory. - Usage of the product. - Encouragement of critical thinking. - User performance assessment.
• **Content**
- Quality of content. - Appropriateness - structure

Table 1: Educational effectiveness category

Following is the description of each aspect as in Stamelos et at (ibid):

Target users' specification: The software packaging or the accompanying reference materials must clearly inform about the approximate age of the target users and about the prerequisite level of knowledge or skills recommended for best use of the software.

Information for topics addressed and learning objectives:

It's very important that instructors and educators are provided with clear and comprehensive information concerning both the topics that the educational software deals with and the learning objectives that it aims to achieve. Obviously the topics addressed by the software must be relevant to the set of learning objectives, so as to enable users to achieve them, and the learning objectives must be appropriate for the target users' age and competence. When educational software is designed for classroom use to ensure that the software is a valuable educational resource, the topics covered and the learning objectives must be compatible with the educational system of the country where the software is used.

Instructional support materials:

They help not only instructors but also users to focus upon the potentialities of the software. They give suggestions on the various teaching strategies instructors can adopt using it in the classroom and inform about how the program can be fitted into a larger framework of instruction, etc.

Adaptation to individual needs:

This aspect has four sub-aspects:

- Feedback: the software product provides feedback information that is not stereotyped, but appropriate for the situation and the users' performance.
- Possibility to follow different learning routes (exploratory learning environments).
- Differentiation of the level of difficulty in respect with the user's performance.
- Level of interactivity.

Strategies for enhancing engagement, attention and memory:

This aspect has the following sub-aspects:

- User motivation which is achieved when the software is able to:
  - Show the users the usefulness of what they learn.
  - Set clear goals and provide indication of how the user is proceeding periodically.
  - Encourage users to envision themselves in an imaginary context or event where they can use the information they are learning.
  - Inspire cognitive curiosity by giving partial information, elements of surprise, stimulating desire to know, etc.
  - Inspire sensory curiosity using sound, visual stimuli, etc.
  - Provide a level of user control, keeping always in mind that too much user control can be detrimental.

  Other characteristics related to user motivation are:
  - Confidence: provide reasonable opportunity to be successful.
  - Competition with the other users (students).
  - Competition with the computer.
  - Competition with the user himself/herself.
  - Competition with the clock.
  - Adjunct reinforcement: Follow the successful completion of any activity with an activity that the user (student) finds enjoyable.
- Varied tasks and activities: the diversity in the way in which the user performs various tasks.
- Retention of information: Retention of information is encouraged when the difficulties are well distributed throughout the program, the topics are clearly connected and summaries of the main topics covered in each preceding section are provided.

Usage of educational software: It's very important to consider the possible usage of the educational software as learning resource in the classroom or by a single user as self-instructional resource, whether it can be useful for the administration of tests, or can be used only for instructor-led tuition.

Encouragement of critical thinking: the degree to which the program provides critical thinking and decision making activities that entail inductive or deductive reasoning and problem-solving skills must be taken into account.

User's performance assessment: For true and actual learning to take place, it is important that the educational software allows the users to constantly monitor and assess their learning progress.

Content

The content of an educational software product has three aspects:

Quality of content: The quality of the content is analyzed with the following attributes:

- Accuracy: measures the absence of inaccuracies in the content presented by the software.
- Clear formulation of the content so as to be easily understandable.
- Completeness: Capability of the software in dealing with all the aspects of each topic.
- Up-to-date.

Appropriateness: This aspect refers to the appropriateness of the reading level for the target users. Users should be able to understand the information presented, so it is essential to check if vocabulary, structure and sentence length are suitable for their level of knowledge, presenting an acceptable degree of difficulty.

Structure: this attribute focuses on the organization of content, which should be logically structured and divided among the sections or modules, in order to help the user to progressively assimilate information.

Evaluation forms:

Some educational institutes or instructors of courses in computer integration in education provide evaluation forms of computer software as. Following are examples of some of these forms that are available on the internet.

- The International Society for Technology in Education[1] provides an evaluation form for software programs[2] . This form includes the 'closed' categories: 'content', 'assessment', 'technical quality', 'instructional design' and other open categories like 'strengths', 'weaknesses', 'the learning strategy' and 'recommendations'.
- O'Neill (1999) whose course 'Computer Applications in Education' is on the internet[3] and the evaluation form, which is part of the course material and has twenty eight items[4].
- Waynesville R-VI Schools[5], which have an evaluation form that includes thirty items[6].

It's worth noting that the form that O'neill (1999) suggests is partitioned into categories that constitute the components of the evaluation, while the form that Waynesville Schools suggest doesn't show the categories that it's composed of.

Online educational software evaluation form:

---

[1] at http://www.iste.org

[2] at http://cnets.iste.org/teachers/pdf/Appendix_D.pdf

[3] at http://www.iol.ie/~aidancbs/tech/course/

[4] at http://www.iol.ie/~aidancbs/tech/course/evalform.doc

[5] at http://waynesville.k12.mo.us

[6] at http://waynesville.k12.mo.us/Media/forms/software%20eval%20form.pdf

It's worth to consider an on-line educational software evaluation form[7], where we can choose the appropriate answer from a list of options and the site evaluates the total score. The form evaluates the following software aspects: (1) educational value (2) entertainment value (3) ease of use (4) design features (5) value (6) package integrity.

Now I describe some of the researches concerning how teachers evaluate educational software.

Teachers go to software:

Prescott (2001) describes the work of preservice teachers to construct tools for evaluating educational software: "The student-teachers work in small groups to explore a range of CALL[8] software from a university database. In addition to considering the software within the frameworks of behaviorism, constructivism and socio-cultural theory, they have to construct instruments for the purposes of reviewing and evaluating the software". Prescott (ibid) says that the preservice teachers construct the evaluation instrument after they view on-line examples of evaluation instruments and examine instruments developed by previous cohorts of students. The preservice teachers construct the evaluation instruments making particular use of the work of Hubbard (1988) with respect to evaluating software for computer-assisted language learning. Prescott (2001) reports that Hubbard (1988) emphasizes the following parameters when evaluating software for language learning:

- The novelty of the field;

---

[7] at http://www.vitalknowledge.com/reviews/AcadiaUniversityFall2002/software%20evaluation%20for%20educators_files/software%20evaluation%20for%20educators%20revised.htm

[8] Computer Assisted Language Learning.

- The problem of "skimming" the software (as one might a textbook) and obtaining an overview;
- The complexities of branching and multiple pathways;
- Visual, auditory and animation dimensions.
- Interactional aspects (extent of student control, extent of software response to input and so forth).

Prescott (2001) reports that 'following Hubbard's ideas the student-teachers in this course endeavor to develop evaluation instruments which provide information about how software will help improve a learner's proficiency in the target language. This means that apart from considering the design features and how the software operates, the student-teachers must also take into account what methodological possibilities are offered as well as the approach inherent in the software'.

Working in this environment and these conditions, the preservice teachers develop tools that have five categories:

1. The purpose or the objectives of the software.
2. The category of design features, which includes features of electronic technology such as use of color and sound, highlighting, branching, layout, graphic features and so forth with the emphasis on the features as aids to learning.
3. The software procedure which assesses clarity of instructions, availability of help, use of hints, definitions, use of examples and models.
4. The category of methodology where the software is examined to determine its potential for allowing a teacher to employ a range of methods: individual learner to computer; paired or group activity, whether the software configuration is open to adjustment and can allow some flexibility in technique and utilization.

5. Approach to language instruction in which a key concern in this category is to identify evidence in the software and its operation of any of the approaches studied.

Another study, which is interested in how teachers evaluate software, is that of Moss (2002-2003) in an investigation which took the form of action research. "The research is based upon the responses to two initial questionnaires that were sent out to a sample of schools, including infant, primary, secondary and colleges. The project sent out 225 questionnaires to 39 schools, and 67 teachers responded from 26 of the schools". Moss (ibid) analyzed the data relating to the selection of software written by the responding teachers 'to determine a means of evaluation that would fulfill teachers' requirements as well as ensuring that such evaluations have appropriate foundations in acceptable practice when using ICT'. Moss found that, when evaluating software, 'the teachers in the sample focused principally on the outcome of the teaching and learning and the content which they hoped to deliver. The means by which these tasks were to be achieved played little or no part in the selection process". Moss remarks: "This, therefore, raises questions regarding decisions to use ICT to accomplish their learning outcomes. If teachers are not valuing those aspects of computer-based learning and teaching which make the use of the computer uniquely suitable for a particular learning style or task, why use the computer?".

The evaluation tool that we suggest:

Some researchers suggested evaluating software through answering specific questions as in Buckleitner (1999). Others suggested detailed aspects as in Dobrica and Niemela (2002), Martin and Shafer (1996) and Wheeler (2005). Some researchers were interested specifically in methods to evaluate educational software (We described the methods of Fernandez

(1997), Gerdt et. al (2002) and Stamelos et. al (2000)). We also paid attention to internet forms which could be used to evaluate software and discussed how teachers and preservice teachers evaluate educational software.

Considering the criteria that researchers suggested to evaluate general software programs we find that most of the aspects that they mentioned are common, so it's important to compare the simplicity of these evaluation methods. The method that Martin and Shafer (1996) suggested seems relatively difficult to implement for the various relations and percents which it includes. The methods that Wheeler (2005) and Dobrica and Niemela (2002) suggested seem very similar and both have sub-categories, but Wheeler evaluations methods seem more detailed and easier to implement, especially for teachers, so we recommend to use these methods when teachers or preservice teachers are involved.

Regarding the evaluation of educational software, we recommend to use the educational part of the methods of Stamelos et. al (2000); i.e. the educational effectiveness aspect, for it covers, in a detailed way, the whole spectrum of educational targets and goals that could be associated with a software program. To do a quick evaluation of a software program or an evaluation that doesn't need deep scrutinizing it's recommended to use the online forms suggested above.

It's recommended to examine further how teachers and preservice teachers evaluate software programs and how they implement Wheeler's methods for evaluating general software programs and the educational part of Stamelos et. al's methods. This allows us to examine how much practical and workable the previous methods are for teachers and to suggest improved methods that could be implemented more easily by teachers.

References:

Buckleitner, W. (1999). The state of children's software evaluation - yesterday, today, and in the 21st century. *Information Technology in Childhood Education Annual*, 211-220. Retrieved August 24, 2005, from http://www.childrenssoftware.com/evaluation.html.

Dobrica, L. and Niemela, E., (2002). A survey on Software Architecture Analysis Methods. Retrieved August 20, 2005, from http://www.cis.ksu.edu/~dag/740fall02/materials/740f02presentations24.pdf

Fernandez, A., (1997). Evaluation of Computer Based Learning Materials. *UniServe Science News, Volume 7*. Retrieved August 25, 2005, from http://science.uniserve.edu.au/newsletter/vol7/fernandez.html.

Gerdt, Miraftabi and Tukiainen (2002). Evaluating Educational Software Environments. *International Conference on Computers in Education*, Auckland, New Zealand. Retrieved July 25, 2005, from http://csdl.computer.org/comp/proceedings/icce/2002/1509/00/15090675.pdf

Hubbard, P. L. (1988). Language teaching approaches, the evaluation of CALL software, and design implications. In W. M. Flint-Smith (Ed.) *Modern Media in Foreign Language Education*. NTC: Illinois. pp227-254

Martin, R. and Shafer, L., (1996). Providing a Framework for Effective Software Quality Assessment. The 6th *Annual International Symposium of INCOSE: Systems Engineering: Practices and Tools*. Retrieved July 23, 2005, from http://www.mitre.org/work/tech_transfer/pdf/risk_assessment.pdf.

Moss, P., (2002-2003). An assessment of the ways in which teachers evaluate software. *ICT in Schools Research and Evaluation Series*. Retrieved August 24, 2005, from http://www.dfes.gov.uk/ictinschools/uploads/docarchive/No.%2016%20Bursaries.doc.

O'Neill, R., (1999). MCE Course Support Materials. Retrieved August 25, 2005, from http://www.iol.ie/~aidancbs/tech/course/

Prescott, (2001). Developing evaluation instruments for CALL software and English second language websites with pre-service English second language teachers. *ITMELT (Information Technology & Multimedia in English Language Teaching) Conference*. Retrieved August 24, 2005, from http://elc.polyu.edu.hk/conference/papers2001/prescott.htm.

Stamelos, I.; Refanidis, I.; Katsaros, P.; Tsoukias, A.; Pombortsis A. and Vlahavas, I.., (2000). An Adaptable Framework for Educational Software Evaluation, in Eds: S. Zanakis, G. Doukidis, I. Anagnostopoulos, *Recent Developments and Applications in Decision Making*, Kluwer Academic Publishers, pp. 347-360. Retrieved July 25, 2005, from http://citeseer.ist.psu.edu/cache/papers/cs/26696/http:zSzzSzdelab.csd.auth.grzSz~katsaroszSzEdSoftwareEvaluation.pdf/stamelos00adaptable.pdf.

Wheeler, D., (2005). How to Evaluate Open Source Software / Free Software (OSS/FS) Programs. Retrieved August 20, 2005, from http://www.dwheeler.com/oss_fs_eval.html .

خلاصة:

أحد أهداف هذا المقال هو وصف تجارب مهمة لتقويم وتحليل برامج حاسوبية بشكل عام وبرامج حاسوبية تربوية بشكل خاص. أدوات التحليل الموصوفة تحتوي على أدوات مفتوحة وأدوات مغلقة على شكل استمارة, ومن ببنها الاستمارة الانترنتية التي تحسب وحدها مجموع النقاط التي يستحقها البرنامج الحاسوبي حسب تقويمك أنت. هدف آخر للمقال هو وصف تجارب سابقة لمعلمين وطلاب–معلمين في تقويم وتحليل برامج حاسوبية. هذا الوصف سيفيد المعلمين والطلاب–المعلمين في اختيار أدوات ملائمة لتقويم وتحليل أي برنامج حاسوبي يحتاجونه لاستعمالهم الشخصي أو لاستعمال طلابهم, كما سيفيدهم في الاستفادة من تجارب معلمين وطلاب–معلمين سبقوهم بتقويم وتحليل برامج حاسوبية تربوية.

תקציר

אחת המטרות של מאמר זה היא לסקור נסיונות חשובים להערכת וניתוח תוכנות מחשב באופן כללי ותוכנות מחשב חינוכיות באופן ספיציפי. כלי ההערכה כוללים כלים פתוחים וכלים סגורים בצורת טופס, ומבין הטפסים טופס מכוון אשר מחשב את הנקודות המגיעות לתוכנה לפי ההערכה שלך. מטרה שנייה היא לתאר מחקרים שעוסקים בהערכת מורים ומתכשרם להוראה לתוכנות מחשב. סקירה ותיאורים אלה יעזרו למורים ומתכשרים להוראה לבחור בכלי אחד או יותר כדי להעריך ולנתח את התוכנות הדרושות להם בשימושם האישי ובשימוש בכיתה לצורך הוראתם. סקירה ותיאורים אלה גם יועילו למורים ומתכשרים להוראה בכך שיחשפו אותם לשיטות שונות של עמיתיהם להעריך ולנתח תוכנות מחשב חינוכיות.