AN ADAPTABLE FRAMEWORK FOR EDUCATIONAL SOFTWARE EVALUATION

I. Stamelos[1], I. Refanidis[1], P. Katsaros[1], A. Tsoukias[2], I. Vlahavas[1] and A. Pombortsis[1]

[1]Dept. of Informatics, Aristotle Univ., Thessaloniki, 54006, Greece
{stamelos, yrefanid, katsaros, vlahavas, pombortsis}@csd.auth.gr

[2]LAMSADE-CNRS,Univ.Paris-IX,Dauphine,Paris Cedex 16, France
tsoukias@lamsade.dauphine.gr

Abstract: This paper proposes a framework for educational software evaluation based on the Multiple Criteria Decision Aid methodology. Evaluating educational software products is a twofold process: both the educational and the technical aspect of the evaluated products have to be considered. As far as the product educational effectiveness is concerned, we propose a set of attributes covering both the general educational features and the content of the product. From the technical point of view, a software attribute set based on the ISO/IEC 9126 standard has been chosen together with the accompanying measurement guidelines. Finally, an evaluation example involving three commercial educational software packages for mechanics is presented.

Key words: Software Evaluation, Educational Software, MCDA

INTRODUCTION

Evaluating software products is a particularly difficult process because many, often contradictory, attributes have to be taken into account. An important effort for defining a universally accepted model has been undertaken by the International Standard Organization (ISO), which has published the ISO/IEC 9126-1, 9126-2 and 9126-3 standards. ISO proposes six attributes, which characterize the quality of a software product: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability* and *portability* [1]. These attributes can be further analyzed in lower-level attributes.

However, the ISO standards do not cope with software attributes that are appropriate for assessing product quality from a non-technical point of view. In the case of educational software it is generally accepted that it is very difficult to develop a predefined set of standards according to which the educational value of the software can be measured. The reason is that each educational software product does not necessarily serve the same learning objectives and the same target users (age, level of knowledge or skills). Therefore, the set of attributes to be chosen for assessing the educational value of a software product must clearly prescribe the evaluation context in each case.

This paper presents an adaptable evaluation framework for educational software products based on the Multicriteria Decision Aid methodology (MCDA) [6, 9], which is suitable for evaluation problems where many attributes must be taken into account. Since evaluating educational software is a twofold process, concerning both the technical and the educational aspect of the evaluated products, the proposed framework consists of two top-level attributes, one concerning the technical features of the evaluated products and one concerning the educational effectiveness of them. The ISO/IEC 9126 standard is chosen as the basis for evaluating the quality of a software product from the technical point of view, while an adaptable set of attributes is proposed for assessing the educational value of the product. A real example concerning the evaluation of three commercial educational software packages is presented. The overall evaluation model used in the example is illustrated and critical points are discussed

In the following section we present the principles of software evaluation using MCDA methodology. Next we present the attributes used for the technical aspect of the evaluation and subsequently we present the attributes selected for the educational aspect of the evaluation. We illustrate this framework with a real example and finally we conclude the paper and pose future directions.

MULTIPLE CRITERIA DECISION AID METHODOLOGY (MCDA)

An evaluation problem solved by MCDA can be modeled as a 7-ple $\{A,T,D,M,E,G,R\}$ where [9]:

- A is the set of alternatives under evaluation in the model
- T is the type of the evaluation
- D is the tree of the evaluation attributes
- M is the set of associated measures
- E is the set of scales associated to the attributes
- G is the set of attributes constructed in order to represent the user's preferences
- R is the preference aggregation procedure

In order to solve an evaluation problem, a specific procedure must be followed [4]:

Step 1: *Definition of the evaluation set A:* The first step is to define exactly the set of possible choices. Usually there is a set A of alternatives to be evaluated and the best must be selected. The definition of A could be thought as first-level evaluation, because if some alternatives do not fulfill certain requirements, they may be rejected from this set.

Step 2: *Definition of the type T of the evaluation*: In this step we must define the type of the desired result. Some possible choices are the following:

- choice: partition the set of possible choices into a sub-set of best choices and a sub-set of not best ones.

- classification: partition the set of possible choices into a number of sub-sets, each one having a characterization such as good, bad, etc.
- sorting: rank the set of possible choices from the best choice to the worst one.
- description: provide a formal description of each choice, without any ranking.

Step 3: *Definition of the tree of evaluation attributes D*: In this step the attributes that will be taken into account during the evaluation and their hierarchy must be defined. Attributes that can be analyzed in sub-attributes are called compound attributes. Sub-attributes can also consist of sub-sub-attributes and so on. The attributes that can not be divided further are called *basic attributes*. An example of such an attribute hierarchy is shown in figure 1.

It should be noted that there exist mandatory independence conditions, such as the separability condition, and contingent independence conditions, depending on the aggregation procedure adopted.

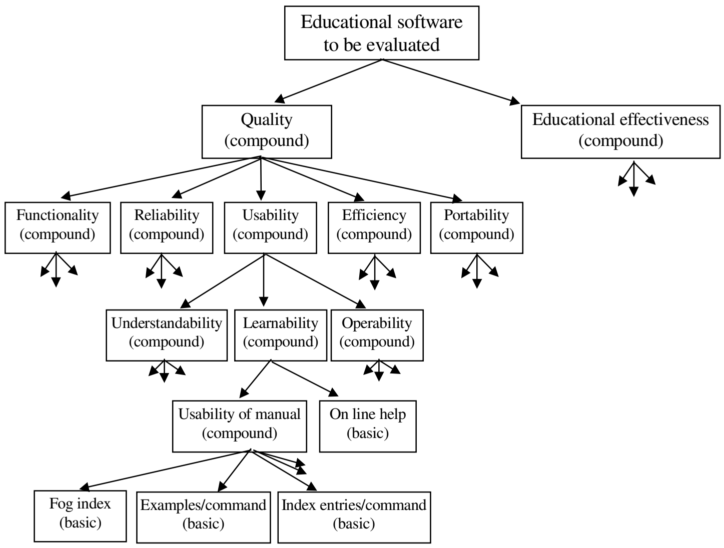

Figure 1. Example of an attribute hierarchy

Step 4: *Definition of the set of measurement methods M*: For every basic attribute d we must define a method M_d that will be used to assign values to it. There are two kinds of values, the *arithmetic values* (ratio, interval or absolute) and the *nominal values*. The first type of values are numbers, while the second type are verbal characterizations, such as 'red', 'yellow', 'good', 'bad', 'big', 'small', etc.

A problem with the definition of M_d is that d may not be measurable, because of its measurement being non-practical or impossible. In such cases an arbitrary value may be given, based upon expert judgment, introducing a subjectivity factor. Alternatively, d may be decomposed into a set of sub-attributes d_1, d_2, … d_n, which are measurable. In this case the expression of arbitrary judgment is avoided, but subjectivity is involved in the decomposition.

Step 5: *Definition of the set of measurement scales E*: A scale e_d must be associated to every basic attribute d. For arithmetic attributes, the scale usually corresponds to the scale of the metric used, while for nominal attributes, e_d must be declared by the evaluator. Scales must be at least ordinal, implying that, within e_d, it must be clear which of any two values is the most preferred (in some cases there are different values with the same preference). For example, for d = 'operating system', e_d could be [UNIX, Windows NT, Windows-95, DOS, VMS] and a possible preference could be [UNIX = Windows NT > Windows-95 = VMS > DOS].

Step 6: *Definition of the set of Preference Structure Rules G*: For each attribute and for the measures attached to it, a rule or a set of rules have to be defined, with the ability to transform measures to preference structures. A preference structure compares two distinct alternatives (e.g. two software products), on the basis of a specific attribute. Basic preferences can be combined, using some aggregation method, to produce a global preference structure.

For example, let a_1 and a_2 be two alternatives and let d be a basic attribute. Let also $m_d(a_1)$ be the value of a_1 concerning d and let $m_d(a_2)$ be the value of a_2 concerning d. Suppose that d is measurable and of positive integer type. In such a case, a preference structure rule could be the following:

- *product a_1 is better than a_2 on the basis of d, if $m_d(a_1)$ is greater than $m_d(a_2)$ plus K, where K is a positive integer*

- *products a_1 and a_2 are equal on the basis of d, if the absolute difference between $m_d(a_1)$ and $m_d(a_2)$ is equal or less than K, where K is a positive integer*

Step 7: *Selection of the appropriate aggregation method R*: An aggregation method is an algorithm, capable of transforming the set of preference relations into a *prescription* for the evaluator. A prescription is usually an order on *A*.

The MCDA methodology consists of a set of different aggregation methods, which fall into three classes. These are the *multiple attribute utility methods* [2], the *outranking methods* [9] and the *interactive methods* [8]. The selection of an aggregation method depends on the following parameters [9]:

- The type of the problem
- The type of the set of possible choices (continuous or discrete)
- The type of measurement scales
- The kind of importance parameters (weights) associated to the attributes

- The type of dependency among the attributes (i.e. *isolability*, *preferential independence*)
- The kind of uncertainty present (if any)

Notice that the execution of the steps mentioned above is not straightforward. For example, it is allowed to define first D and then, or in parallel, define A, or even select R in the middle of the process.

ATTRIBUTES FOR EVALUATING THE TECHNICAL FEATURES OF EDUCATIONAL SOFTWARE

According to ISO 9126 the technical aspect of quality is decomposed into six sub-attributes and each one of them is further decomposed in sub-sub attributes. Quality is decomposed as follows:

– Functionality [suitability, accuracy, interoperability, compliance, security]

– Reliability [maturity, fault tolerance, recoverability]

– Usability [understandability, learnability, operability]

– Efficiency [time behavior, resource utilization]

– Maintainability [analyzability, changeability, stability, testability]

– Portability [adaptability, installability, conformance, replaceability]

ISO 9126 basically prescribes a general framework, which may be adapted to the characteristics of a specific evaluation problem. For specific types of educational software, some of the above attributes may be irrelevant. In the example presented in the next section, which concerns commercial multimedia software for personal use, maintainability and portability have been considered of no interest. In the following sub-sections, we discuss briefly the above attributes.

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 e.t.c. It can be decomposed in five sub-attributes:

- *Suitability* is the degree of presence of a set of functions for specified tasks.
- *Accuracy* is the degree of provision of right or agreed results or effects.
- *Interoperability* is the degree to which the software is able to interact with specified systems (i.e. physical devices)
- *Compliance* is the degree to which the software adheres to application-related

standards or conventions or regulations in laws and similar prescriptions.

- *Security* 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 e.t.c.).

Reliability

Reliability is defined as the capability of the software to maintain its level of performance under stated conditions for a stated period of time. It can be decomposed in three sub-attributes:

- *Maturity* 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* is the ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
- *Recoverability* is the capability of software to reestablish its level of performance and recover the data directly affected in case of a 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 can be decomposed in three sub-attributes:

- *Understandability* is the user's effort for recognizing the underlain concept of the software. This effort could be decreased by the existence of demonstrations.
- *Learnability* is the user's effort for learning how to use the software.
- *Operability* is the user's effort for operation and operation control (e.g. mouse support, shortcuts e.t.c.).

Efficiency

Efficiency is the relationship between the level of performance of the software and the amount of resources used, under stated conditions. It can be decomposed in two sub-attributes.

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

Maintainability

Maintainability is defined as the effort needed to make specified modifications. It can be decomposed in four sub-attributes.

- *Analyzability* is the effort needed for diagnosis of inefficiencies or causes of failure or for identification of parts to be modified.
- *Changeability* 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

Portability is the ability of the software to be transferred from one environment to another. It can be decomposed in four sub-attributes:

- *Adaptability* is the software's opportunity for adaptation to different environments (e.g. other hardware/OS platforms).
- *Installability* is the effort needed to install the software in a specified environment.
- *Conformance* is the degree to which the software adheres to standards or conventions related to portability.
- *Replaceability* is the opportunity and effort of using the software in the place of specified older software.

ATTRIBUTES FOR EVALUATING THE EDUCATIONAL EFFECTIVENESS

In contrast with the technical aspect of the evaluation, there is no broadly accepted model for assessing the educational effectiveness of a software package. The reasons for this are mainly:

- It is very hard to describe the context of all possible educational software evaluation problems with a single attribute framework. For example, the evaluation carried out by a teacher or a trainer is a completely different problem compared to the evaluation process carried out by a decision-maker of an educational institution. In addition, factors that must be taken into account are the type of target users the evaluator has in mind while undertaking the evaluation and the way he or she intends to use the software (for example, to teach a specific topic, or to enhance students' understanding of a certain topic).

- There are several types of educational software products. According to [3] these types are: '*drill and practice*', '*tutorials*', '*simulations*', '*instructional games*' and '*problem solving*'. Each one of these software types may need different evaluation attributes.

- An educational software product may have such original characteristics that prevent the use of a predefined set of evaluation attributes.

In this work we tried to take into consideration all elements relevant to teachers, trainers, parents and users. The proposed 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 proposed framework is based on the work presented in [7], which has been modified by removing the attributes related to the technical aspect of the evaluation and by extending in more detail the attributes related to the educational purposes of the evaluation. According to our approach the educational effectiveness attribute of a software product is decomposed in two sub-attributes, each one of them being further decomposed. The first two levels of this analysis are shown in table 1. In the following sub-sections we will discuss these attributes in more detail.

Table 1. Educational effectiveness decomposition

• educational features
- target users specification - information for the topics addressed and the 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

Educational Features

– *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 the topics addressed and the learning objectives*: It is 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 the 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 education system of the country where the software is used.

- *Instructional support material*: Another aspect to take into account when evaluating the educational features of a particular piece of software is the quality of the instructional support material it provides, either in print and/or as printable files from disc or on-line resources. In fact, they can significantly help not only instructors but also users to focus the potentialities of the software, giving suggestions on the various teaching strategies instructors can adopt using it in the classroom, informing about how the program can be fitted into a larger framework of instruction etc.

- *Adaptation to individual needs*: This attribute is further decomposed in four sub-attributes:

  - *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)*: It is important that the software allows the users to follow different learning routes through the program.

  - *Differentiation of the level of difficulty in respect with the user's performance*

  - *Level of interactivity*

- *Strategies for enhancing engagement, attention and memory*: This attribute is decomposed further on in the following sub-attributes:

  - *User motivation*: User motivation is achieved when the software is able to:

    - Show to the users the usefulness of what they learn.
    - Set clear goals (e.g. number of questions that need to be completed without a mistake) 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 e.t.c.
    - Inspire sensory curiosity using sound, visual stimuli e.t.c.
    - 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 other users (students)
- Competition with the computer
- Competition with the user him/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 & activities*: The diversity/motonony in the way performing 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 is 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 onto 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 is measured according to three sub-attributes:

– *Quality of content*: The quality of the content is analyzed in the following lower level 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 attribute 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.

A REAL WORLD EXAMPLE

This section presents an example in which three commercial educational software packages for personal use are compared. The packages concern mechanics for high schoolers and have been selected from the Greek market. We are not presenting the names of the evaluated products, since this would not increase the worthiness of the example, but we will refer to them with the terms *P1*, *P2* and *P3*.

In the following sub-sections we will present first the attributes used for the technical part of the evaluation, together with their measurement scales and the ratings of the evaluated products for the various basic attributes. Next we will present the attributes used for the educational part of the evaluation. Finally we will present the aggregation procedure and the way the final result was obtained.

Technical part of the evaluation

As mentioned before this part of the evaluation is based on the quality scheme of ISO 9126. Concerning *Functionality*, we removed attributes *Interoperability* and *Security*, since our specifications required neither capabilities for exchanging real experimental data with physical instruments nor user authentication. Moreover, we removed accuracy, since our packages do not perform computations that require high degree of accuracy. We have further decomposed *Suitability* in three sub-attributes: *Theory*, *Experiments* and *Exercises*. For *Theory* we used as a metric the number of *Subjects* covered by each package. For *Experiments* and *Exercises* we used as a metric the ratio of *Experiments*/*Subject* and *Exercises*/*Subject* of each package.

Concerning *Reliability*, we considered only its sub-attribute *Maturity*, which expresses the frequency of faults due to software problems. *Recoverability*, *Fault tolerance* and *Availability* have been considered irrelevant. We assigned an increased weight to *Reliability*, since for educational software this kind of faults is unacceptable.

Usability is decomposed in *Understandability*, *Learnability* and *Operability*. We analyzed *Learnability* in terms of *Usability of manual* and *Availability of help functions*. Finally we analyzed *Operability* in *Availability of installation program*, *Message clearness* and *Cancelability ratio*.

Concerning *Efficiency*, we considered both *Time behavior* and *Resource utilization*. In order to 'measure' the *Time behavior*, we used the subjective impression of the package's response times. Concerning resource utilization, we considered the requirements of the package in memory, disk space and CPU type and speed.

Finally, we considered *Maintainability* and *Portability* as being irrelevant for commercial multimedia software for personal use.

Table 2 presents all the attributes used for the technical part of the evaluation, together with their hierarchy, the measurement scales for the basic attributes, the weights assigned to the attributes and the values assigned to the three alternatives for the basic attributes. In case of arithmetic basic attributes, we define also a threshold, which represents the minimum difference that must exist between two alternatives, in order to consider the one superior to the other. In case of nominal basic attributes, specific procedures can be defined for assigning values to the alternatives, or expert judgement may be used.

Table 2. The sub-model for the technical part of the evaluation

	Attribute	Weight	Scale	Threshold	P1	P2	P3
1	Functionality	3					
1.1	Suitability	2					
1.1.1	Theory	4	Number of sections	2	14	15	13
1.1.2	Experiments	2	Experiments/Section	2	2.1	2.8	1.9
1.1.3	Exercises	2	Exercises/Section	3	4	8	5
1.2	Compliance	1	{high, average, low}		high	high	aver.
2	Reliability	9					
2.1	Maturity	1	{high, average, low}		high	high	high
3	Usability	5					
3.1	Understandability	2	{high, average, low}		high	high	high
3.2	Learnability	2					
3.2.1	Usability of manual	1	{high, average, low}		aver.	high	aver.
3.2.2	Help functions	2	{complete,partial,missing}		partial	compl	partial
3.3	Operability	1					
3.3.1	Installation program	1	{available, missing}		avail	avail	avail
3.3.2	Message clearness	2	{high, average, low}		high	high	high
3.3.3	Cancelability ratio	1	{complete, partial, missing}		miss	partial	miss
4	Efficiency	2					
4.1	Time behavior	1	{good, average, bad}		good	good	good
4.2	Resource utilization	1					
4.2.1	Memory	1	MB	16MB	8	8	8
4.2.2	Disk space	1	MB	10MB	17	15	21
4.2.3	CPU	1	Type/Speed	50 MHz	P100	486/33	486

Educational part of the evaluation

For the educational part of the evaluation we used all the attributes presented in the previous section. Table 3 gives the detail for the attributes used, their weights, the measurement scales and the values assigned to the three packages for the various basic attributes. In this case there are not arithmetic basic attributes, so no thresholds have been used.

Table 3. The sub-model for the educational part of the evaluation

	Attribute	Weight	Scale	P1	P2	P3
1	Educational Features	4				
1.1	Target users specification	0.5	fully > partially > missing	missing	partial	partial
1.2	Information for the topics	1	fully & consistent > fully but not consistent > partially > missing	partial	fully & consist.	fully & consist.
1.3	Instructional support material	0.5	complete > adequate > missing	missing	missing	missing
1.4	Adaptation to the individual needs	2				
1.4.1	Feedback	1	differentiated > stereotyped > missing	missing	differ.	missing
1.4.2	Different learning rules	1	possible > not possible	not	not	not
1.4.3	Differentiate the level of difficulty	1	possible > not possible	not	not	not
1.4.4	Level of interactivity	2	good > moderate > bad	moder.	moder.	moder.
1.5	Strategies for enhancing engagement	2				
1.5.1	User motivation	2	good > moderate > bad	bad	bad	bad
1.5.2	Varied tasks	2	varied > monotonous	monot.	monot.	monot.
1.5.3	Retention of information	1	good > moderate > bad	bad	bad	bad
1.6	Usage of educational software	1	many cases > one usage	one	many	many
1.7	Encouragement of critical thinking	2	existent > missing	missing	missing	missing
1.8	User's performance assessment	1	many types > one type > missing	one	many	one
2	Content	6				
2.1	Quality of content	5				
2.1.1	Accuracy	1	accurate > inaccurate	accurat.	accurat.	accurat.
2.1.2	Clear formulation	2	clear > not clear	not	clear	not
2.1.3	Complete	1	complete > incomplete	incomp	compl.	incomp
2.1.4	Up-to-date	1	up-to-date > relatively old > old	to-date	to-date	to-date
2.2	Appropriateness	3	appropriate > not appropriate	approp.	approp.	approp.
2.3	Structure	2	modular > linear > unstructured	modul.	modul.	modul.

Aggregation phase

Having constructed the two evaluation sub-models, we have combined them into a single hierarchy, assigning weight value 4 to the technical sub-model and weight value 6 to the educational effectiveness one. Taking into account the fact that many of the basic attributes are nominal and that there exist weights and thresholds, we selected to use ELECTRE II [5]. This is an outranking MCDA method that provides a complete or partial ordering of equivalence classes from the best ones to the worst ones, considering also ties and incomparable classes. ELECTRE II calculates an ordering relation on all possible pairs of alternatives and uses an exploitation procedure to construct a preference relation among them, without providing any notion about their absolute distances. ELECTRE-II demands the definition of a concordance threshold, which in general indicates in what percentage of the attributes one alternative should outperform another (taking also into account the weights), in order that the former is considered superior to the latter. In our case, we set the concordance threshold to the value $c = 0.6$.

Since the computation of the ordering relations of ELECTRE is a hard process to be done manually, we used EPS [10], an Expert System for Software Evaluation that supports evaluation with various MCDA methods. We entered the model together with the values assigned to the three alternatives, we selected ELECTRE-II (this was also the suggestion of the system) and the system extracted the final result. According to this, the three alternatives are ordered as:

$$P2 > P1 = P3$$

CONCLUSIONS AND FUTURE WORK

In this paper an adaptable framework for educational software evaluation is proposed, which takes into account both the technical and the educational aspect of this type of software products. For the technical part of the evaluation the ISO 9126 standard is adopted. For the educational part of the evaluation, it seems that it is not possible to define a single set of attributes appropriate for any problem. The attribute framework to be used depends on the type of target users the evaluator has in mind, on the way he or she intends to use the software and on the instructional strategy that has been chosen. Although a quite general set of attributes based on the ideas of [7] has been proposed, it seems to be more important to support the adaptation of the proposed set of attributes to the specific circumstances of an evaluation problem.

We have applied the framework in the comparison of three commercial educational software products for personal use. The evaluation has been performed with the Multiple Criteria Decision Aid methodology, which is suitable for evaluation problems where many attributes have to be taken into account. The evaluation process has been supported by ESSE, an Expert System for Software Evaluation, which embodies several MCDA methods, together with knowledge for various types of software evaluation problems.

In the future we will continue working on the proposed framework, applying it in a sufficiently large number of cases. Moreover, we plan to explore the applicability of more MCDA methods, such as other outranking and multiple attribute utility methods, some interactive methods, etc. Finally, we will explore the applicability of these methods to other categories of software evaluation problems, obtaining additional experience.

REFERENCES

[1] ISO/IEC 9126-1, Information Technology - Software quality characteristics and sub-characteristics (1996).
[2] Keeney R.L. and Raiffa H., Decision with multiple objectives, John Wiley, New York (1976).

[3] Lockard J., Abrams P. and Many W., *Microcomputers for educators*, Little, Brown and Co., Boston (1987).
[4] Morisio M. and Tsoukias A., IusWare, A methodology for the evaluation and selection of software products, *IEEE Proceedings on Software Engineering,* 144, 162-174 (1997).
[5] Roy B. and Bertier P., La methode ELECTRE II - Une application au media planning, *in OR72*, M. Ross (ed.), North Holland, Amsterdam, 291-302 (1973).
[6] Roy B., *Multicriteria Methodology for Decision Aiding*, Kluwer Academic, Dordrecht (1996).
[7] Severino A.U., Educational Effectiveness Evaluation Criteria, *Emerging New Technologies in Education*, Symposium, Samos, Greece (1998).
[8] Vanderpooten D. and Vincke P., Description and analysis of some representative interactive multicriteria procedures, Mathematical and computer modelling, no.12, (1989).
[9] Vincke P., *Multicriteria decision aid*, Wiley, New York (1992).
[10] Vlahavas I., Stamelos I., Refanidis I., Tsoukias A. (1998), *ESSE: An Expert System for Software Evaluation*, Knowledge Based Systems, Elsevier, 4 (12), 183-197 (1999).