Accessible software for early maths education: focusing on content, strategies, interface

G. Dettori\textsuperscript{a} & M. Ott\textsuperscript{b}

\textsuperscript{a} Istituto per la Matematica Applicata, C.N.R., Genova, Italy
\textsuperscript{b} Istituto per le Tecnologie Didattiche, C.N.R., Genova, Italy
dettori@ima.ge.cnr.it ott@itd.ge.cnr.it

ABSTRACT

Students presenting learning disabilities may have trouble accessing some educational tools, including software. In this paper we analyse what accessibility might mean when referred to content, strategies and interface of educational software. We point out how a very precise analysis of these components should be made in order to allow learning-disabled pupils to make good use of them. Based on this discussion, we conclude that the accessibility of a software application does not only depend on its characteristics, but also on how it is used in relation with each pupil's learning objectives and current cognitive level, as well as the whole educational setting.

1. INTRODUCTION

In the wide range of disabilities affecting school students in the developmental age, learning difficulties are statistically significant (CCLP, 1998). If not correctly tackled from the beginning, they can evolve into real handicaps, leading to early school dropout and difficult integration into the labor world. Learning-disabled pupils obviously need particular attention and care, but do they also need special tools? In this paper we try to answer this question, as regards educational software. We therefore discuss which features should be considered, and from what point of view, when deciding to use a software tool in a remediation program for learning-disabled pupils.

Effectiveness and functionality are certainly major objectives (Hines 1994) to pursue when choosing software products to use with such children; this means choosing tools which, if properly applied, can actually contribute to attaining the proposed educational objectives. Effectiveness and functionality, however, seem to be strictly connected with accessibility: no significant educational goal may be reached if we rely on tools (no matter how powerful, rich and appealing) which prove difficult to use for these particular pupils, and that result in overlooking their specific, individual needs from the point of view of cognitive abilities (Simon & Halford, 1995), emotional/behavioural attitudes, competence already developed.

We are currently carrying out a long-term experimental project on the use of educational software in mathematics education, with elementary school children (2nd to 5th grades) that have specific learning difficulties (underachievers but not cognitively impaired). In this framework, the software has been used both for functional diagnosis and as a remediation tool. This experience led us to reconsider the issue of software accessibility for learning-disabled pupils; we analysed more closely than is usually done in the literature the nature of the main aspects of educational software tools, that is, content, educational strategy and interface.

These considerations might prove useful not only for mathematics, but also for the other basic abilities, such as learning to correctly read and write, whose lack marks from the beginning pupils who are unable to keep pace with the rest of their class. The experience developed in the above-mentioned project, together with previous experiences in the fields of dislexia or disgraphia (Morchio et al, 1989), suggest us the consideration that with learning-disabled students it is particularly necessary to make a very precise analysis...
of content, educational strategy and interface, since even small shades of differences can result into cognitive obstacles for these particular students.

2. SOFTWARE CONTENT

When a superficial analysis of a software tool is performed, the content appears easy to determine, simply by examining the user's manual or the set of exercises proposed. However, this is totally misleading since the cognitive task underlying the solution of arithmetic exercises actually differs according to how they are formulated.

Let us consider, for example, the case of programs that drilling pupils in addition operations. An exercise requiring computation of the sum of two numbers (e.g., \(4+3=\ldots\)) aims to develop reckoning abilities, and suggests a view of the operation as a procedural process. If the result of the sum is to be chosen from a (small) set of possible results (e.g., \(4+3=\{5;7;2;10\}\)), the exercise lends itself rather to the development of evaluation capabilities (or lets the pupil guess rather than reason!). If an addition is to be performed using three given numbers (e.g., write an operation using the following symbols: \(4, 7, 3, =, +\)) or choosing three numbers from a given set, then the accent is rather on emphasizing the relational character of addition. If the result is shown and one of the addends is missing (e.g., \(4+?=7\)), the relational value of addition can be exploited to introduce the concept of difference. Though they all concern the same operation, these exercises actually have a different knowledge content and present different conceptual difficulties. The resulting impact on students' learning and cognitive development will differ depending on which of these exercises they tackle. This may not be obvious to a non-mathematician tutor, nor is it usually explained explicitly in the user's manual –and we wonder if it is always clear even to most software designers! Though the relational view of the operation usually proves to be more difficult to grasp than the procedural one, whether these different views of sums are equally accessible to a pupil strongly depends on his/her personal cognitive characteristics and on the way they are introduced. A careful mentor, paying attention to the different levels of understanding of the different forms of operation, can focus more precisely on each pupil's cognitive gaps (Fuchs et al., 1989).

![Fig.1 - The Rooster exercise in the “MathMax” software package (Dainamics)](image)

Moreover, not all exercises concerning operations on numbers are actually focused on reckoning abilities. An interesting example is that shown in Figure 1, where the user has to arrange the numbers by shifting each row left or right, so that the numbers in the column under the rooster sum up as assigned. In this exercise, making computations is secondary; what is actually exercised is the capability to make plans in order to reach the expected result. It is not surprising, then, that some children in our experimental group who were able to handle such simple sums without difficulty found it particularly arduous. On the other hand, learning to plan a solution strategy is an important part of mathematics learning, since it is necessary for developing problem solving abilities: we were not surprised, then, to find that the same pupils who
found difficult in working out a solution strategy for the above exercise or in learning one from their mentor turned out to be poor problem solvers. Hence, it is also important to consider what kind of solution strategies a program allows or supports, and by means of what kind of tools. Though strategies are often built-in features of software tools, teachers should not feel compelled to accept them passively, since it is possible to have some control through interaction with the pupils by guiding them to make their reasoning processes explicit.

A third important component of mathematical content knowledge is the construction of meanings for concepts and symbols (Carey, 1991). It must be clear to the teacher that being able to apply the computation technique correctly for some operation does not imply understanding its meaning, and this is witnessed by the number of pupils who have learned to perform operations but still try to guess which one needs to be applied when solving a problem. None of the above mentioned exercises concerning sums offers any tool to build the meaning of the addition operation, which can be constructed and understood only in a problem solving (i.e. applicative) environment, possibly initially disregarding the use of arithmetical symbols in order to avoid conceptual overloading (Carreter, 1988).

Hence, from the point of view of software content, accessibility essentially means proposing exercises and activities concerning the very shades of knowledge that the pupil has been prepared - and is ready - to learn or to rehearse, bearing in mind that by themselves learning-disabled pupils are hardly able to make connections between different aspects of the same mathematical object, and understanding that a slight difference in point of view, if not suitably addressed, can hinder the pupil from making use of an apparently valid learning experience.

3. EDUCATIONAL STRATEGY

Educational strategy refers to how a software tool leads the user to acquire some content knowledge. It is deeply embedded in the design of each educational program and is a key point to be checked in order to make any educational software effective with respect to some educational objective (Dreyfus, 1993).

Most software tools currently available for early maths education follow a drill-and-practice approach, that is, offer a predetermined set of exercises which consist in answering a question or solving some operation, but do not involve the construction or exploration of meanings. Such programs can help pupils to become fast and fluent in performing some operations but have little impact on their understanding of the concepts at play if meanings are not previously constructed and supported by means of suitable explanations and activities. Other programs, on the other hand, show a hands-on, constructivist approach (Norman, 1996), or make widespread use of representations as support to concept acquisition, or offer communication tools to favor learning as a shared process within a group of pupils. What makes one or other approach preferable is usually the nature of the educational objective (Hammond, 1992): simulation-based activities lend themselves to exploration and understanding of meanings, while repetitive ones aim to fix and automate purely algorithmic skills such as summing up numbers or memorizing multiplication tables.

Recent trends in mathematics education research tend to privilege educational strategies based on personal construction and discovery (Nardi, 1996), rather than the drill-and-practise one, which nevertheless is the type most often implemented by commercial software tools simply because it can be more easily automated.

In our experience, however, we have noticed that some repetitive tasks can prove to be as challenging as explorative ones for learning-disabled children; working out a solving strategy to tackle simple computations, or building a mental model to support table memorization, can turn out to be an exploration task as well. The same exercise, suitably introduced by the teacher, can be equally used to make a child grow in his/her mathematical thinking or to fix things (which is not necessarily a simple task for these particular pupils!). What makes the difference is whether such activity interacts with the child's proximal development zone (Vygotsky, 1986), that is, it is ready to be understood under teacher's guidance, or has already been understood and its knowledge must be consolidated, and this cannot be determined a priori but only evaluated by the mentor through careful observation of the child's attitude and behaviour.
Hence, we can say that a software is accessible from the point of view of the embedded educational strategy if the child is in a condition to make use of it to grow cognitively. As with the case of content, strategy accessibility depends not only on the software itself, but also on the use that the teacher is planning, or able, to make of it with respect to the educational objective and the current cognitive needs of the pupil.

4. INTERFACE

The interface determines to what measure the users are put in a condition to make good use of the above-mentioned components, in that it concerns how the content is structured and presented. It not only determines how appealing, understandable and easy to use a program will be, but can also support thinking development, e.g. by means of a variety of meaningful representations, or by allowing a stimulating and non-constraining dialogue, hence possibly proving to be a concrete tool of cognitive growth. It is natural then to wonder what accessibility means as concerns the interface.

Though every pupil obviously has his/her own difficulties and cognitive style, the drawbacks which are usually shared by all these children are: scarcity of attention and memory, difficulty in understanding assignments, difficulty in conceptualizing, that is, building suitable mental representations to support mathematical operations as well as organizing the acquired knowledge at metacognitive level by recognizing analogies and connecting related concepts. Hence, based on a broad analysis of the literature and on our own experience, we think that interfaces accessible to learning-disabled children should present in wider measure those characteristics of consistency, essentiality and clarity which are anyway desirable for educational software.

Therefore, features which can hinder learning appear to be:

- poor logical organization, which makes the connection between learned concepts or abilities difficult;
- overly "nice" graphical presentation, rich in gadgets devoid of any meaning functional to the solution activity, which distracts, rather than helps, pupils perform the cognitive task at hand: even apparently easy tasks often require a big mental effort of these children, who welcome any possibility to switch from attentive work to play;
- proposing representations which are too symbolic to be easily understood, or purely descriptive, and useless for supporting an operative understanding of concepts;
- assigning tasks which require too much time to be solved, since the attention span of learning-disabled children is usually limited;
- joining too many concepts or abilities to be learned in one exercise, which results in an excessive cognitive effort;
- requiring to work fast, or not giving children enough time or chances to work out a correct answer, which makes them lose pace and prevents them from learning, since organizing knowledge into mental representations and retrieving it requires time, especially for children who are unable to do so instinctively.

Some features, on the other hand, appear to ease learning:

- assigning tasks which make sense in relation with the life experience of the pupil; one of the main difficulties with learning mathematics is the centrality of abstraction, which increases more and more with the level of complexity; adding more abstraction at task level can be too much for these children;
- making use of different representation registers (Duval, 1995) for the same assignment (such as symbols, words, graphics, sound), in order to convey the same information in different, connected ways;
- giving meaningful feedback for every answer; this gratifies the pupil when the answer is correct, and helps pinpoint problems when it is not.

Finally, it ought to be remembered that no software is so versatile as to really adapt its functioning to the needs of every pupil, hence a tutor's intervention is always necessary to introduce, explain, graduate and combine tasks. This is even more necessary with pupils who are hardly able to adapt their cognitive activity to a program's functioning.
5. CONCLUSIONS

Students presenting learning disabilities may have trouble in accessing some educational tools, including software. Accessibility in this case is to be understood as linked not to physical but to cognitive barriers, since we are not dealing with a physical problem. However, similarly to what happens with material situations, software should be considered accessible when pupils are in a condition to make good use of it.

When a software tool is used in a remedial program for learning-disabled pupils, attention should not be limited to the interface, since software accessibility is also strictly connected to the content and educational strategies embedded in the product. These three aspects are strictly interconnected and the issue of making each of them accessible cannot be suitably tackled independently of the others. The accessibility of content and educational strategy of software tools does not depend only on their peculiar characteristics, but also on the use that the teacher is planning, or is able, to make of them with respect to the educational objectives and the current cognitive needs of the pupil.

ACKNOWLEDGEMENTS

The experience mentioned has been carried out within the SVITA projects involving CNR (IMA and ITD), ASL3 Genovese and Provveditorato agli Studi di Genova.

REFERENCES


