Designing software to maximize learning[1]

Bridget Somekh

Scottish Council for Research in Education

This paper starts from the assumption that any evaluation of educational software should focus on whether or not, and the extent to which, it maximizes learning. It is particularly concerned with the impact of software on the quality of learning. The paper reviews key texts in the literature on learning, including some which relate directly to software development, and suggests ways in which a range of learning theories can inform the process of software design. The paper sets out to make a contribution to both the design and the evaluation of educational software.

Introduction

I take it as axiomatic that those involved in developing educational software intend that it should maximize learning. The evaluation of educational software, therefore, needs to focus mainly on this central issue. There are still, however, two ways in which 'maximization' can be measured: either in terms of the increase in *the amount of learning*, or in terms of the increase in *the quality of learning*. In either case, an important contributory factor in the evaluation of educational software is the amount of take-up in terms of purchase and use. The crude indicator of BSH (number of Bums on Seats for number of Hours) is of some value, because if software is not used it cannot have any impact at all on learning. But it is an indicator which should only be used with caution because badly designed software used frequently presumably does less to maximize learning than well designed software used infrequently.

I am concerned here with the design of software which maximizes the quality of learning. I will also deal briefly with the issue of sensitivity to the context of use, but the wider problem of the take-up and use of software, although important and interesting, is beyond the scope of this paper. The paper is based on a review of key texts in the literature on learning, including some which relate directly to software development. I hope it will make some contribution to both the design and the evaluation of educational software.

Learning theories

There is no one recognized right way of thinking about learning. Theories change over time and tend to reflect the attitudes and technologies of the period. The origin of the Jesuits' emphasis on learning by heart lies in the scarcity of books in medieval times. This was later developed into a concept of 'training the mind' in which the habit of, and capacity for, memorization was seen as a prerequisite for scholarship. Such a theory of learning had direct pay-offs. With so much information literally at the tip of their tongues, Jesuit scholars became known for the acuity of their thinking and their administrative acumen, as well as their rigorous approach to matters of faith.

In the twentieth century, the main debate has been between those following Skinner, who see learning as 'conditioning' by means of a carefully planned sequence of 'stimuli' designed to elicit a set of desired 'responses', and those following the ideas of Vygotsky and Bruner, who see knowledge as 'a process rather than a product'(Bruner, 1966, p.72), constructed by the learner as a result of experiences, critical reflection upon those experiences, and social interaction (e.g. discussion) (see Prawat, 1991). The debate is clearly influenced by new technology which has greatly reduced the need for the memorization of facts, although the need to 'train the mind' continues in the long-running debate on the need to develop 'transferable skills'. Desforges (1989) rejects the confrontational stance between behaviourists and constructivists and sees a need for two kinds of learning. In his view, there are times when we need to learn *apparently* arbitrary facts (e.g. the sequence of elements in the periodic table), and there are times when we need to learn concepts (e.g. the concept of valence and co-valence in the molecular structures which make up those elements). Although, as in this case, there is often an interconnection between these two kinds of knowledge, the learning process is quite different. As Desforges says: 'The latter is knowledge constructed by human intelligence in interaction with and adaptation to the environment and unlikely ever to be understood unless reconstructed by the learner' (p.20).

Much early software of the drill-and-practice kind was designed according to Skinnerian principles to support the acquisition of apparently arbitrary facts. Underpinning this software is the notion that it is of no consequence that the periodic table is not in fact arbitrary – the simplest way of embedding it as a tool that the learner can use with confidence is to teach it *as if it were* arbitrary. There is still a great deal of software that works on the basic principle of teaching facts and information as if they were unproblematic and then testing the students to see if they can put these facts and information to use at a fairly simple level to answer a question. It is very much more difficult to devise software which supports 'constructive' learning whereby concepts are internalized by the student and contribute from then onwards to his or her capacity for intuitive problem-solving. (It is also much more difficult to assess this kind of learning.)

It follows that there are two key issues for software developers:

1. deciding on the right balance of emphasis between the two kinds of learning (given that one is much easier to achieve than the other – and therefore cheaper);
2. once this decision is made, devising computer-mediated experiences capable of supporting the second kind (constructive learning of concepts).

The rest of this paper tries to throw light on the second of these issues. Before moving on, however, it may be useful to explore what is meant by 'constructive' learning in a little more depth. MacDonald and his colleagues (MacDonald *et al*, 1976) provide some insights from their evaluation of the NDPCAL programme, a large-scale initiative funded by the Department of Education and Science in the 1970s. Their five-scale typology of the levels of learning via CAL contains 'operational definitions', as follows:

Key features of the types of interaction are:

Type A interactions: recognition:
Text-dependent; require matching of superficial features of information presented to information previously presented; non-productive.

Type B interactions: recall:
Text-dependent; superficial engagement of student with content; reproductive; combinatorial manipulation of syntactical and logical features of text.

Type C interactions: reconstructive understanding or comprehension:
Text-independent but discourse-dependent; involve semantic interaction with content; reconstructive; productive; involve comprehension of statements, concepts, or principles.

Type D interactions: global reconstructive or 'intuitive' understanding:
Experiential learning; discourse-independent; focus on structure of discipline; student as master of discipline; problem-solving; conservative of discipline.

Type E interactions: constructive understanding:
Student 'creates' fields of knowledge; discipline-independent; exploratory; problem-finding; domain-dependent.

In terms of computer-software design, MacDonald *et al* distinguish interactions of Types D and E from the lower levels. Type C interactions lie on the borderline between the lower levels and higher levels: they represent the lowest level of constructive understanding.

In the case of interactions of Types D and E, more complex opportunities are provided for the student to demonstrate learning, and evaluation of whether or not learning has occurred will require a far more complex judgement: the correctness or incorrectness of the response cannot be decided in terms of a simple discrepancy between response and text. Thus, in these two types of interaction, the student demonstrates learning by much more complex acts of meaning production than in Types A, B and C.

Research in Artificial Intelligence has focused on the attempt to design computer-user interactions at levels D and E. Some progress has been made, but the problems remain rather intractable. For one thing, such interactions tend to require that the software has the ability to accept open-ended responses in natural language and respond to them appropriately. Some of the most interesting AI software has been developed to have the capacity to learn the user's working style and level of prior knowledge, in order to tailor the interactions to suit better the user's needs. Work of this kind has been incorporated in Intelligent Tutoring Systems, but the ability of such systems to promote constructive learning is not always clear, and their production costs are very high.

The MacDonald typology provides a useful monitoring device which a development team might use during the design stage. There are two ways in which software developers can focus their energies on ensuring level C 'constructive' learning interactions, and may be able to move towards promoting the higher D and E levels:

1. Linking on-screen interactions with learning away from the computer.

Both in NDPCAL and in more recent software development research (see Harding, 1974; 1993), one fruitful way of extending CAL interactions to level C and beyond has been by linking them with activities undertaken away from the computer.

2. Use of simulations (using technological features such as graphical user-interfaces, high-resolution graphics and hypertext navigation).

MacDonald *et al* point out that simulations have the potential to stimulate a student's thinking at level D by 'present[ing] the student with inferences from his [*sic*] responses, and thus creat[ing] cognitive disequilibrium' (p.12).

Mind maps or mental schema

The theory behind constructivist learning is that each individual develops mental schema or 'mind maps' which serve to inform future thinking or action. These schema are fundamental to the way we understand all experience. As babies we begin to build up schema which enable us to distinguish a human face from its background. More abstract conceptualization involves the same process of constructing a meaning and pattern from a jumble of sensory information. These schema then enable us to function with confidence in a complex environment. As Bruner puts it (1966, p.2): 'Much of perception involves going beyond the information given through reliance on a model of the world or events that makes possible interpolation, extrapolation, and prediction.' Effective learning depends on the creation of new schema, or on existing schema being revised, extended or reconstructed.

The mind-maps theory has two implications for software development:

1. In order to learn, individuals need to be able to build on their existing schema.

This points to a need for individualizing learning. Learners need to exercise control over their learning so that its pacing and direction are guided by questions arising from their developing understanding. At a superficial level this can be achieved by giving the user control over what to learn and in what order by offering choices. Since icon buttons have replaced the rather cumbersome system of layers of menus, it is now much easier to make this process appear genuinely exploratory and attractive to the user. However, at a deeper level it is arguable that software cannot address this need unless it contains the ability to 'model the user' (see above in the short discussion of Artificial Intelligence). Neither the language, nor the complexity of the investigation of the concepts, nor assumptions about the learner's prior knowledge, can be adapted to the learner by offering choices in this way. A 'pre-test' which will act to select appropriate material for a particular user may be the best way of attempting to address this problem.

2. In order to learn, individuals need support in constructing new schema.

This suggests the need for structure underpinning the software design. Navigation is the centrally most important issue in developing software in a hypertext environment (Laurillard, 1987). Learners build their mental schema on 'where they go' in the software and 'what they do'. Structure is largely a matter of sequencing the presentation of information or concepts, identifying sections, establishing links between sections and setting up a system for reviewing the structure both as a whole and in parts. A book provides such a structure (more or less well) through its table of contents, divisions into chapters, headings and sub-headings, etc. (An indication that we use the structure of text in a book to help develop mental schema is that we are often able to visualize the layout of a page when wishing to check on information.) In software development, structure is created for the learner through the approach taken to the key navigation issues of user-control and information sequencing.

Plowman (1992) has identified a problem with interactive multimedia that the learner's control over the 'path of disclosure' obscures the underlying structure which the authors have given the material. To some extent, therefore, there is a conflict between the simple solution of giving users choice as a means of individualizing their learning experience as appropriate to fit their existing schema, and the need to provide them with sufficient structure to enable them to adapt this as the basis of new schema. In practice, there is a need for a balance: there needs to be structure *and* choice, but the latter must be restricted as far as this is necessary to prevent the former from being unhelpfully obscured.

An aid in constructing mind maps is to have advance knowledge of the ground to be covered (what Ausubel and Robinson, 1969, p.145, call 'an advance organizer'). There is then much greater likelihood that the individual can develop meaningful schema which link, where possible, with their existing schema. Students are sometimes given an 'advanced organizer' in the form of an introductory lecture to a course or a set of learning objectives. In the same way, hypermedia or multimedia software can be used to present graphical overviews to serve as advance organizers.

However, the process of developing schema is never mechanistic. Providing an advance organizer, allowed by a series of learning experiences designed to cover each aspect of the conceptual field in turn, will not be sufficient. Learning does not proceed incrementally along a linear path of increasing difficulty. Opportunities for revision are important because it is unreasonable to expect students to remember everything they have learned on a future occasion. They need to revisit concepts, perhaps many times, until they are securely laid down in well developed mental schema capable of enabling intuitive decision-making. Bruner (1960, pp. 52–4) addresses this need through his design for a 'spiral curriculum' in which learners are introduced to a series of concepts, each one in a number of different ways, over a period of time (specifically through different kinds of content involving slightly different approaches). This principle may be a relatively easy one to adopt in software design, although in practice designers usually rely on students using the software many times without variation – and with the risk of monotony.

Another aid to developing schema which software can incorporate is a Buzan-style on-screen note-making facility. This allows students to record the web of linkages which reflect the meaning they are constructing from using the software. Given the facility continuously to extend and adapt this note-web, students can record their developing

mental schema of the material, with the option of adding hand-written notes to the print-out at a later stage.

Scaffolding

Learning for Vygotsky and Bruner is essentially a social phenomenon. It depends to a considerable extent on the ability to use cognitive tools such as language and other symbol systems. In Vygotsky's theory, learners have limits to their current level of achievement, beyond which they are unable to go without the interventions of a teacher or peer. With the help of such interventions, learners can go beyond these limits into their 'zone of proximal development' (Vygotsky, 1986, p.187), and such experiences serve to extend their understanding so that next time they may be able to achieve this level of understanding without such mental and social 'scaffolding'.

It is claimed that interactive software can provide a kind of mental scaffolding. At the lowest level, it can provide the incentive to continue to work on-task for longer periods without the intervention of a teacher. This may be little more than the plate-spinning phenomenon, whereby the student gets feedback on progress and is set a new task (although the mere setting and answering of questions may not achieve even this low-level effect because of its dullness).

As a result of this recognition of the social nature of learning, many writers place considerable emphasis on the importance of discussion in facilitating learning. Partly in recognition of this, and partly because shortage of hardware has dictated it, computer use in UK schools has become largely a group activity since the advent of the microcomputer in the early 1980s. Many university students will therefore be used to this way of working which does appear to have many advantages. If software is designed for use by students without peer group support, it is important for development teams to bear in mind that interaction between learner and software must be livelier than it might otherwise need to be, because it is replacing, and not supplementing, learner-teacher or learner-learner interactions.

Situated learning, simulations and microworlds

Recently many researchers have placed emphasis on the need for learning to be 'situated' to enable the construction of schema (Brown *et al*, 1989). This theory suggests that much learning is made more difficult because of the context of learning – i.e. ideas have to be grasped through an abstract representation, part of which may be more to do with the context of learning (e.g. copying notes from the blackboard before the lecturer rubs them off) than the context of application. In contrast, when it is situated, the meaning of what is being learned is directly supported by the context of learning. An example might be learning physics in a research laboratory.

Some software provides an element of situated learning because it simulates an authentic situation, or alternatively provides a microworld in which the learner's experience is authentic within the frame of an alternative environment (for an early definition see Papert, 1980). Everything in the simulation or microworld is contained in a computer model which has an integrated set of rules and provides an integrated experience. In that

sense the metaphor and content are fused. Sometimes the microworld involves the user in learning the microworld's language (e.g. natural language and syntax which the software accepts). Always it involves the user learning the limits of the model. The metaphors which control content, simulation, interface and programming structure all contribute to the total effect of the microworld.

Nevertheless, there must always be a limit to the authenticity of computer-mediated experience. Computer simulations by their very nature are presented through a computer interface. In terms of the theory of situated learning they may be very unhelpful if they necessitate the user learning too complex a set of rules governing the computer context of the simulation and unrelated to the simulated 'authentic' context.

One of the most important features of situated learning is the idea of learning from expert colleagues who model appropriate behaviours and approaches to problems. If software has a strong emphasis on problem-solving and exploration, there is little room for expert modelling. The learner is on his or her own. This can be overcome by including some examples of problem-solution by posing problems and offering the option for a step-by-step solution with explanations, including graphical displays. This has been tried with some success by Harding (1994, p.78). The danger of offering expert models, however, is that the user may learn formulaic responses instead of understanding the concepts.

Another important feature of situated learning is that what is learned can be applied immediately in a real situation to solve a problem. Apart from the reservation about the intervening computer interface, this is where simulation software can be particularly useful.

Meta-cognition

Since 1983, Salamon (1992) has used the construct AIME (Amount of Invested Mental Effort), or 'mindfulness', in his work on learning. When writing about the use of computers to support students' self-study he says:

> While the open-endedness of tools is an opportunity for the awakening of mindful engagement and opportunity to sustain it, one has already to be mindfully inclined to take these opportunities. If young writers would seriously and intentionally attend to the expert-like guidance of the Writing Partner, or to the way variables affect each other in the systemic models of STELLA, they'd learn quite a bit. But, alas, only the already mindfully inclined, the ones who find thinking a pleasant challenge and who do not avoid the expenditure of mental effort, are the ones to intentionally expend the effort needed to notice the guidance or the inconsistency in the model's structure. (p.13)

At the lowest level, mindfulness is dependent upon motivation. But mindfulness is essentially a conscious cognitive activity which can be developed by means of critical self-reflection on oneself as a learner. For example, it is quite possible for an individual to have established habits of passive learning (acquired from prior training) so that he or she is unaware of the need to construct knowledge through interrogating ideas and trying to develop mental schema. Becoming aware of this is the first step for these individuals in becoming mindful, and therefore more effective, learners. This has led writers like De Corte (1990, p.73) to see a role for educational software in 'enhancing the acquisition of

meta-cognitive skills and learning strategies through explication of, and reflection on [the learner's] knowledge (deficiencies and misconceptions versus strengths) as well as on their thinking methods and learning activities (powerful versus weak)'.

It is fairly easy to build in explicit strategies to enhance meta-cognition. For example, pre-tests can include some element of analysis of an individual's learning style which can be fed back to the user. Another low-cost solution is to provide an on-screen note-book of the type mentioned above for recording meta-cognitive reflections. This can be very effective if the student is subsequently required to use the notes in discussion with other students or the lecturer.

Motivation

Motivation is essential to learning. Bruner (1966) points out that human beings are the only animals which depend on passing learning on from one generation to the next. We learn for a future purpose and not simply as a response to natural curiosity. This necessitates a large element of delayed gratification in learning, so that we cannot depend on natural curiosity alone to motivate learners. One common strategy for software developers is to present the user with a microworld which appeals to the imagination in some way as well as providing a simulated context for a range of human-computer interactions. Now that the technology is becoming more advanced, there is increasing scope for microworlds to be delightful and absorbing and, at least initially, to have a strong impact on user-motivation.

To move a student from initial motivation to sustained engagement with the task is difficult unless the student has already developed a capacity for sustained engagement. Salamon's 'mindfulness', which Bruner calls 'the will to learn', originates for most of us in a track-record of enjoyable and successful learning experiences. Natural curiosity has in this way been nurtured to produce intrinsic motivation capable of sustaining our interest over a long period of delayed gratification. Motivation is heavily dependent on the learner's prior experience and attitudes. Students are likely to bring with them a range of emotional blocks which undermine their confidence, reduce their ability to remain on-task for any length of time, and in this way obstruct their learning (what Pirsig, 1974, pp.298–306, calls 'internal gumption traps' or 'value traps which block affective understanding'). Some of these may relate directly to the computer itself as the context of learning (in this sense no computer-mediated learning can ever be 'situated'). Many students may have had negative experiences of computer use, and others may perceive themselves as non-technology people so that using a computer threatens their own sense of self-worth grounded in their self-image (Somekh and Davis, forthcoming). Software developers can try to overcome this by designing microworlds which counteract any hard technological image, but this will never be wholly satisfactory because microworlds carry with them their own social and value sets which are unlikely to be in tune with all users – or even whole categories of users.

'Flow' or cognitive engagement

One of the great advantages of books, films, lectures, or any form of presentation which is uninterrupted, is that a learner's mindfulness can change gear into an intense level of

concentration known sometimes as 'flow' (Csikszentmihalyi, 1982), and sometimes as 'cognitive engagement' (Kozma, 1991). In this state, the learner may lose all sense of the context of learning (time and place), the pace of learning accelerates, and there is greatly enhanced motivation which spills over into a sense of excitement and continuing reflection after the event. Turkle (1984) gives many examples of how computers can induce this kind of intense concentration. In some of these, it is associated with the development of repetitive psycho-motor skills, employed under time pressure; in others, it is associated with the intellectual excitement of high-level cognition and creativity resulting from programming in non-linear languages such as Logo. However, computer use can have very different consequences. For example, interactive software is unlikely to induce flow if the learner's thought processes are interrupted too frequently by the need to carry out tasks imposed by the computer.

Translation (or transformation) between symbol systems

One of the main problems in any learning is the need to 'translate' concepts from a symbol system, such as language or numbers or graphical representations/models (Bruner, 1966), into meaningful mental schema. These symbol systems are the basic tools of human intelligence – our ability to use them is what differentiates us from other animals. When learners are asked to imbibe concepts from a textual explanation, they have to make a translation of that explanation in terms of their own experience, based on clues provided by the writer. The writer has attempted to encode a concept in a symbol system (text), and now the reader attempts to decode it and link it to his or her existing mental schema. Content screens can make heavy demands of this kind on the learner.

One of the most difficult problems facing software designers is how to get across sufficient content to enable learners to engage in problem-solving (Dublin, 1988). Until comparatively recently there has been an assumption, originating presumably in the poor quality of VDUs, that the computer is not good at handling large quantities of text. But concepts which are linguistically constructed may be extremely difficult to impart in any other form. A related problem, which I have already touched on, is how to get across sufficient information about a microworld or the rules of a simulation to enable the learner to function intuitively within them. Often such information is given in text form in a set of preliminary screens. With high-quality VDUs and graphical environments, and the sudden increase in screen-displayed texts in CD-ROM and multimedia, it is no longer necessarily true that computer software should be designed with a minimum of text. However, text does still pose a problem if it is to be incorporated on the screen alongside graphics, or if the software is designed for use by a group – which will necessitate a larger font size. The need to reduce the number of words on the screen can lead to truncated text which does not present the concepts adequately. If the main focus of the software is to be some form of investigation or the exploration of a model, and if the concepts can only be delivered by means of considerable quantities of textual explanation, it may be best to produce printed materials to go alongside the software.

The more interactive the software becomes, the more it is possible to allow learners to learn by exploring models, or building models, rather than by digesting text. However, designing software that supports this kind of learning requires a high level of creativity

and painstaking planning. Moreover, there are great differences between individual learners when it comes to the relative difficulty they experience in translating from different symbol systems. Some learners will find it easier to translate from text to their mental schema than from graphical models to their mental schema.

The problem of such translation for learners also includes 'transforming' meaning from one symbol system to another. A typical transformation problem for learners is the interpretation of graphs (Mokros and Tinker, 1987; Smith, 1993). These are usually presented alongside a textual explanation. A graphical representation can communicate a range of detailed information in easily assimilable form, but its interpretation requires a sound understanding of the underlying concepts, e.g. of the likely relationships between the different kinds of information displayed in the graph. There is a marked tendency for teachers (and the producers of textbooks and educational materials) greatly to underestimate the difficulties that novices have with the interpretation of graphs. It is assumed that graphical representation will clarify and explain textual representation. Often it simply overlays it with a second level of difficulty. A common error, which indicates the problem of transformation, is the novice's tendency to 'read' the graph as a picture (i.e. to see points plotted at the top of the page as more likely than those at the bottom of the page to signal a large quantity of something).

Many writers are agreed that the computer offers particular advantages in being able to display graphs dynamically so that the learner can either explore the relationships between the variables by experimentation, or be provided with a dynamic model of the transformation between text and graph (Kozma, 1991); i.e. the user changes the text and the computer transforms this into changes in the graph. Software can also be used to 'proceduralize' understanding by demonstrating how a problem can be investigated (and solved?) step by step, according to clearly defined rules (Kozma, 1991).

Feedback and assessment

There are always considerable problems in assessing learning adequately because cognitive processes are private, hidden and unique to the individual. These difficulties are compounded by the need to integrate assessment within a software package, because this usually precludes the possibility of the student giving natural-language responses to questions. Yet, some element of feed-back or self-assessment is essential to the learning process, and both students and tutors tend to expect computers to provide assessment which records the student's progress and obviates the need to assess the work in some other way.

The main problem of computer-mediated assessment is that it is difficult to prevent testing from emphasizing the lowest A and B levels of the MacDonald typology of interactions – recognition and recall – at the expense of even the C level of reconstructive understanding, simply because the former are so much easier to test than the latter. (An additional problem is that as soon as testing enters the realm of understanding of concepts there is likely to be difference of opinion among lecturers as to the 'correct' answer.) The need to provide a formal assessment option or facility for tracking students' progress for the benefit of providing lecturers with a record at a later stage has the danger of placing too high a constraint on the educational design of the software. It necessitates

the need for computer assessment of higher levels of learning. Such assessment demands considerable sophistication in software design and programming.

Deciding against formal assessment is risky, however, given the deep acculturation of both university students and lecturers into study as an almost wholly assessment-orientated activity. There is the risk that students will view the software as an optional extra if it is not seen as an essential part of the 'exchange of effort for grades' (Doyle, 1979) which keeps them motivated and on-task. There is also the risk that lecturers will not recommend the software to students strongly because they know they will have no way of checking whether or not students have used it, and will therefore not be able to rely on the software as a substitute for any of their existing teaching. If BSH is to be the main performance indicator for software developers, project teams need to adopt a strategy to safeguard against this. One possibility is to produce supplementary, paper-based assessment materials to be used after completion of the modules.

A note on the context of use

Teaching and learning in both schools and universities are highly routinized activities. Lecturers and students expect to behave in certain ways (e.g. working at networked terminals in a computer room), and the structures of institutions are generally supportive of these routines (e.g. money is likely to be allocated to pay specialists to maintain the computer network and to purchase software by licence so that it can run on the network rather than on individual laptops). Software which is designed to suit a different institutional context (e.g. demanding a more sophisticated network, or intended to be used on stand-alone computers as a more integral part of subject study in the science laboratory) poses challenges to these routine behaviours and institutional structures, and as a result it is necessary to provide very explicit guidance for users, and give considerable prior warning before asking individuals to undertake trialling. To guard against these kinds of problem, courseware should probably be designed so that it can be used in more than one kind of learning context.

Note

1 A version of this paper was presented at the conference of the Association for Learning Technology (ALT-C 94), University of Hull, September 1994.

References

Ausubel, D. P. and Robinson, F. G. (1969), *School Learning: An Introduction to Educational Psychology*, London and New York: Holt, Rinehart and Winston.

Brown, J. S., Collins, A. and Duguid, P. (1989), 'Situated cognition and the culture of learning', *Educational Researcher*, January–February, 32–42.

Bruner, J. S. (1960), *The Process of Education*, Cambridge MA: Harvard University Press.

Bruner, J. S. (1966), *Towards a Theory of Instruction*, Cambridge MA and London: The Belknap Press of Harvard University Press.

Csikszentmihalyi, M. (1982), 'Towards a psychology of optimal experience', in Wheeler, L. (ed.), *Review of Personality and Social Psychology,* volume 2, Beverley Hills CA: Sage, pp. 13–36.

De Corte, E. (1990), 'Learning with new information technologies in schools: perspectives from the psychology of learning and instruction', *Journal of Computer Assisted Learning,* 6, 69–87.

Desforges, C. (1989), 'Understanding learning for teaching', *Westminster Studies in Education,* 12, 17–29.

Doyle, W. (1979), 'Classroom tasks and student abilities', in Peterson, P. L. and Walberg, H. J. (eds.), *Research on Teaching: Concepts, Findings and Implications,* Berkeley, CA: National Society for the Study of Education, McCutchan, pp.183–209.

Dublin, M. (1988), 'Individualized instruction on the computer: the meaning behind the myth', *Interchange,* 19 (2), 15–24.

Harding, R. D. (1974), 'Computer-aided teaching of applied mathematics', *International Journal of Mathematical Education in Science and Technology,* 5, 447–55.

Harding, R. D. (1993), 'Computer interactive texts', *in Proceedings of the Fourth International Conference on Technology in Collegiate Mathematics,* Reading, MA: Addison-Wesley.

Harding, R. D. (1994). 'Nuffield interactive mathematics courseware', *Computers and Education,* 22 (1/2), 73–82.

Kozma, R. B. (1991), 'Learning with media', *Review of Educational Research,* 61 (2), 179–211.

Laurillard, D. (1987), 'Computers and the emancipation of students: giving control to the learner', *Instructional Science,* 16, 3–18.

MacDonald, B. and the UNCAL team members (1976), 'The educational potential of computer-assisted learning: qualitative evidence about student learning', paper prepared by UNCAL, the independent educational evaluation of the NDPCAL, CARE, UEA.

Mokros, J. and Tinker, R. (1987), 'The impact of microcomputer-based labs on children's ability to interpret graphs', *Journal of Research in Science Teaching,* 24 (4), 369–83.

Papert, S. (1980), 'Microworlds: incubators for knowledge', pp. 120–34 in *Mindstorms,* Brighton: The Harvester Press.

Pirsig, R. (1974), *Zen and the Art of Motorcycle Maintenance,* London: Bodley Head; reprinted London: Corgi/Transworld.

Plowman, L. (1992), 'An ethnographic approach to analysing navigation and task structure in interactive multimedia: some design issues for group use', in Monk, A., Diaper, D. and Harrison, M. D. (eds.), *People and Computers VII: Proceedings of HCI 92,* Cambridge: CUP.

Prawat, R. S. (1991), 'The value of ideas: the immersion approach to the development of thinking', *Educational Researcher,* 20 (2), 3–10.

Salamon, G. (1992), 'Computer's first decade: Golem, Camelot, or the Promised Land?', invited address to Division C, AERA Conference, San Francisco, April 1992.

Smith, H. (1993), *The Use of Computerized Databases in Critical Inquiry by Pupils Aged 8 to 11 Years*, D. Phil. thesis, University of Sussex.

Somekh, B. (1989), 'The human interface: hidden issues in computer-mediated communication affecting use in schools', in Mason, R. and Kaye, A. (eds.), *Mindweave*, Oxford: Pergamon Press, pp. 242–6; reprinted in Boyd-Barrett, O. and Scanlon, E., *Computers and Learning*, Addison-Wesley in association with the Open University, 1991.

Somekh, B. and Davis, N. (forthcoming), 'Getting teachers started with IT and transferable skills', in Somekh, B. and Davis, N. (eds.) (forthcoming – 1997), *Using IT Effectively in Teaching and Learning: Studies in Pre-service and In-service Teacher Education*, London, New York and Toronto: Routledge.

Turkle, S. (1984), *The Second Self: Computers and the Human Spirit*, London, Toronto and New York: Granada.

Vygotsky, L. (1986), *Thought and Language*, Cambridge MA: MIT Press.