British Journal of Educational Technology *Vol 39 No 2 2008* 251–267
doi:10.1111/j.1467-8535.2008.00815.x

*The current status of instructional design theories in relation to today's authoring systems**

A. Fred O'Neal

At the time of this writing (1986) Fred O'Neal was the Technical Director of the Training Systems Division at Wicat. He is now an interactive systems and instructional design consultant. His ongoing professional, editorial, academic and political activities are focused on training technology and improvement of practice in human performance improvement. Address for correspondence: PO Box 666, Freeland, WA 98249, USA. Email: fred@whidbey.com

Abstract
It is of course very difficult to accurately project important characteristics of the future state of any rapidly evolving field, and the field of authoring systems for computer-assisted instruction (CAI) is no exception. However, strong trends in evolving CAI systems of today would seem to indicate some important characteristics of the software systems underlying tomorrow's automated instructional systems. Under the premise that form follows function, this paper will explore the functional impetus of some of the most important trends extant in terms of their roots in a powerful, emerging technology of instructional design. This technology promises to complement the advancing hardware and software technologies, by supplying that *sine qua non* of successful instructional materials. It is inevitable that systems of the future will make (and indeed are making) significant accommodations of this vital technology as it matures, and therefore some important characteristics of tomorrow's systems can be predicated upon what is already known of this technology of instructional design.

Introduction
One of the most important influences of the new instructional technology stems from the fact that it is a technology. Rooted in a growing body of research prompted by the empirical observations of a large body of practitioners, defined by an increasingly rigorous calculus of practice, and validated by an impressive range and volume of application, the technology has matured to the stage where it is sufficiently well defined to support instructional engineering applications by non-theoretically inclined users.

*This paper was first published in two parts in *Interactive Learning International* Vol. 2, nos 3 & 4, 23–26 and 29–33, by John Wiley & Sons Ltd.

The implications of this degree of definition needs to be explored in terms of tomorrow's authoring systems. One of the questions this paper will address is 'To what extent can (or should) we build today's instructional strategies into CBT authoring systems?'

CBT languages, as they have been developing, although they may be mechanically efficient and intellectually seductive, are essentially instructionally irrelevant. That is, although they provide the tools for defining interfaces between users (authors, students, proctors, instructors, etc.), the content and the systems capabilities, these are at best only definitions of mechanical boundary conditions (Bunderson, 1977). The languages are deficient in the metaphors of the instructional technology; they imply no necessary considerations of primitive content structures, strategy definition, or locus of control. Most are written on a level of discourse far below that which would be desirable (Pask, 1969). In short, whereas they may serve to communicate to the system 'how' to do something, they offer little help to the instructional designer in terms of the more important question, namely 'what should be done' instructionally.

Therefore one important evolutionary step in the CAI systems of tomorrow will be increasingly sophisticated 'authoring systems' as opposed to 'authoring languages'. To be sure, these 'authoring systems' will perforce include authoring languages as an important component, but more significantly, they will include software and data structures for cueing, prompting and tutoring the instructional designer/developer in 'what' he should be doing instructionally, based on the analytical and prescriptive models implied by the design technology. This facet of software development is being actively pursued by several different groups. Current programs, approaches and progress will be discussed in the body of the paper, as will a more detailed description of the types of functions a mature authoring system should include.

Another important influence on the software architecture of coming CAI systems is the trend towards well-defined componentised content structures. The efficiency and economy of authoring which can be realised with componentised content (eg, content structures which are discrete and well defined, such as rules, definitions, examples practice items, introductions, summaries, helps, etc) are attractive. The 'factory approach' to instructional content development and coding is extremely powerful in a componentised environment (Bunderson, 1973).

Another strong influence on tomorrow's systems is made possible by separation of content and strategy. Once basic instructional strategies can be built into the system to deal very directly and efficiently with the classes of instructional problems for which strategies and components have been defined (again, this would most efficiently be done at a higher level of discourse than that of most current authoring languages). VAULT, TICCIT, CDS and WISE represent beginning steps towards this desirable capability. The paper will discuss the nature of componentised content structures and the separation of content and strategy, and examples of system software accommodations, which might be implied, will be described.

With componentised content structures and generally definable strategies, it becomes possible for tomorrow's CAI systems to negotiate or manipulate the locus of instructional control as never before (O'Neal, 1973).

Definition of the software structures needed will require development of both state and trait 'advisors' and models of the learner which will emerge as CAI systems mature. Some exist now, empirically determined, as the basis of powerful adaptive models (ie, the basic mathematical skills materials developed at Stanford, etc), but if any degree of learner control is to be allowed, the software structures must be capable of functioning in a communications and tutor mode, as well as the more familiar executive function, controlling student progress.

Finally, the automated instructional systems of tomorrow must more effectively deal with the fact that not all instruction takes place on the system itself. The artificial lines between CAI and CMI (computer-managed instruction) will be broken down and a full set of management functions (both for managing instruction and for managing the development of instruction) will be necessary as part of any system.

Implications of the emerging instructional design technology for authoring-system development will be outlined using examples from existing systems' capabilities to provide perspective.

A brief history of instructional design

The history of instruction reveals three different periods or stages of development, which recapitulate the history of science: the artistic approach, the empirical approach and the systematic approach. In the artistic approach, the design, delivery and evaluation depends upon intuition and personal experience. To the extent that one person's intuition or experience was better than another's, better instruction ensued. This approach to instructional development, can generally be characterised as having a 'cottage industry' flavour. That is, when it comes to the development of a given segment of instruction the author becomes a jack of all trades. He does the analysis, design writing, sometimes the design of graphics, and test developments. He approaches each segment as a completely new task and employs an idiosyncratic or artistic approach to this 'new' instructional development problem.

The model of 'instructional developer as artist' has several serious deficiencies. First, it is inefficient, in that much time is spent inventing the new approach to each segment. Secondly, art is difficult to evaluate or teach. Quality is variable, and the managers of CAI curriculum development activities find it difficult to train and maintain the required number of artists. Finally, artists are, alas, temperamental. Their interest in the new art (CAI) is often very temporary. They have difficulty tolerating extended periods of routine activity such as inputting instructional sequences. And, by definition, their output is difficult to predict or control. These factors lead to uneven rates of production, inability to meet deadlines, and several other undesirable results. They have been a major cause of the current state of the bibliography of CAI materials, which includes hundreds of demonstrations, clever games and interesting segments, but pitifully few integrated, full courses or even modules of instruction.

This pot-pourri may be good for demonstrating the range of possible applications of CAI, but it has not had a major effect on the progress of education and training and has not allowed CAI to take its place as a major contributor to the education and training communities. In addition, it puts an incredible load on the CAI student who must continually learn and relearn the 'rules of the game' while passing through lessons generated using an infinite variety of approaches and styles.

In the empirical approach, intuition is used to formulate hypotheses, which are then used to design experiments and collect data. Instructional methods or techniques which are supported by the data tend to be accepted over those which do not.

The empirical approach to institutional development evolved from the programmed instruction industry and emphasises carefully stated objectives, course try-out and revision based on data from testing students. It has some advantages over the more artistic approaches. It does put some of the onus for poor student performance on instructional developers, and it is likely to eventually lead to more effective courseware. However, it too has problems. The emergence of this empirical model was not accompanied by the development of carefully developed strategies or prescriptive models for getting from well-stated objectives to first drafts of the courseware, or from revision data to final versions. Such strategies and models are left to the individual developer with a few exhortations to use small steps, overt responding and feedback. This model is often prohibitively expensive in CAI applications because it results in long linear sequences, the need for several revisions and many of the same problems associated with the artistic approach described earlier.

A curriculum-development activity which uses task analysis, behavioural objectives and formative evaluation and revision techniques is still essentially using a cottage industry' approach when the carefully stated behavioural objectives are turned over to an instructional developer who operates according to an intuitive or artistic model in order to develop his instructional displays. Among the greatest misuses of CAI have been the applications best described as 'automated textbooks' and 'automated programmed instruction sequences'. Seldom has the very special potential of CAI been so grossly ignored as in these applications.

In the systematic approach, analytic procedures or methods are used to design, develop and evaluate instruction. These procedures are derived from a combination of refined experience, accumulated research findings and learning theory.

The systematic approach is called instructional systems development (ISD) or just simply instructional development. ISD has emerged in response to the pressing need for a more effective, efficient and reliable educational system. It involves things such as task analysis, behavioural objectives, criterion reference testing, individualised instruction, formative evaluation and media selection. Although ISD was originally developed in the context of military training, its use has spread to all domains of instruction and it is probably the dominant instructional methodology existing today.

Some important characteristics and benefits of ISD are as follows:

Characteristics
1. Based on research, experience and intuition;
2. Procedural/explicit;
3. Documented;
4. Evolutionary and adaptable;
5. Potential for rigour;
6. Potential for automation.

Benefits
1. Validity;
2. Reliability;
3. Efficiency;
4. Productivity;

1. Documented, procedural approach that allows use of differentiated staff (leverage)
2. Simplifies staff training
3. Improves quality control
4. Allows job aiding
5. Allows automated support.

A major potential of ISD is that instructional activities (design/development/evaluation) can be carried out by technicians or automated via computers once the appropriate procedures are developed and tested. For example, a media selection algorithm can be applied (by a technician or computer) to decide the appropriate delivery medium for a particular instructional sequence. Or an individualised instructional program can be carried out, not by the Socratic interaction between tutor and student, but by a CBT course or by a course manager and a computer-managed instruction system. Similarly, test items can be constructed not only by an experienced teacher, but also by an item-generation rule which creates items with the behaviour, conditions and criterion specified in any objective.

Instructional strategies

One of the most important by-products of ISD has been the emergence of a research and experience-based set of focused and proven instructional strategies based on classification of each instructional or training objective according to the class of learning/teaching problem it represents. Considerable gains in productivity can be achieved when the designer takes advantage of the fact that in a 1000-objective course there are not 1000 unique and different teaching/learning problems. For example, if the designer successfully develops an instructional strategy to teach the objective 'state the name, location and function of the controls and indicators on the high power illuminator radar', he probably will not have to invent a totally new and different instructional strategy to address the objective 'state the name, location and function of the controls and indicators on the FM communications receiver'. It has been this author's experience in developing instruction and training for over 10 000 objectives in education, the military and industry that 85 to 90% of the cognitive objectives encountered could be successfully dealt with using a basic library of less than 12 instructional/training strategies.

To the extent that a basic library of instructional strategies can be created and refined then, one of the most important activities in the instructional design process becomes the correct classification of each instructional or training objective in terms of the type of learning/teaching problem it represents and the instructional strategy that this implies.

Towards this end, a variety of taxonomic schemes for classifying instructional objectives has emerged. Some of the more notable of these include Bloom's taxonomy of cognitive objectives, Gagne's levels of learning and Merrill's taxonomy of instructional objectives. Some of these taxonomies have already provided the basis for prescriptive instructional strategies for incorporation into elaborated ISD models. Gagne's levels of learning provide a rough basis for the library of instructional strategies first designed by the Training Analysis and Evaluation Group (TAEG) of the US Navy, and included the Interservice Procedures for Instructional Systems Development (IPISD) used as the basis for training design by the US military. The chief of naval education and training for the US Navy has adopted an ISD variant using Merrill's taxonomy and elaborated as a very refined and fully documented and supported set of instructional strategies, which are described and taught in the Author Training Course (O'Neal, Faust & O'Neal, 1979) developed under Defense Advanced Research Projects Agency (DARPA) funding in the mid-1970s. This taxonomy was also used by the National Science Foundation team at the University of Texas and Brigham Young University as a partial basis for the instructional specification of the initial authoring software for the NSF TICCIT system.'

Which instructional taxonomy you use should depend upon your requirements. For example, it is certainly the case that if a taxonomic scheme with 10 categories can successfully classify 90% of your training objectives in terms of the type of training problem they represent (and therefore identify the instructional strategy which would best address that problem), a more elaborated taxonomic scheme with 30 categories could probably classify 97 or 98% of your training objectives. But is it worth it? True, use of the simpler scheme will require that your instructional design staff 'invent new solutions' to 10% of your training objective problems. Perhaps this could be alleviated by the creation, for this project, of a small number (three or four perhaps) of 'new' objective/strategy types for this project. The alternative, namely that of using the 30-category taxonomy, although it may require invention and validation of no new instructional strategies, has its own costs. For one thing, the instructional design and development staff must be taught to use a more complex taxonomic scheme in categorising the training objectives. In addition, once objectives have been correctly classified, the design staff must be trained in the correct implementation and use of 30 alternative instructional strategies instead of the 10–14 implied by the simpler taxonomic scheme. Having been faced with this problem numerous times in contexts where a few instructional designers had to provide input and guidance to a large number of subject matter experts and course developers, this writer has always found it to be more efficient to use the simpler taxonomic approach. A caveat should be introduced here however. Improvements in efficiency and productivity using any taxonomic approach are dependent upon the degree to which the instructional strategies associated with each taxonomic category have been explicitly documented and supported with procedures, job aids and training in the implementation of the strategy.

The DARPA Author Training Course referenced earlier is an example of a relatively mature strategy library. The course consists of an eight-volume learner-controlled, individualised training course in the taxonomy and its use, and in the implementation of each instructional strategy implied by the taxonomy. This instruction uses the instruction strategies of the taxonomy, where appropriate, to teach the materials. In addition to the complete instruction on each strategy, there are elaborated job aids complete with check lists, clarifying notes and references to the instructional materials for use by instructional developers in implementing and instructional strategy required for each training objective as it is encountered in the development process.

To the extent that the instructional strategy library for any taxonomic scheme is documented and supported, the implications for CBT authoring systems of the future should be clear. If a step-by-step procedural job aid supported by individualised instruction and help can be created to aid in the correct application of the instructional strategy by CBT developers on real-world development projects, then clearly this strategy could be incorporated as a 'prototype' or menu structure in a sophisticated CBT authoring system.

What is an authoring system?

Today's 'authoring systems' for CBT address a very small part of the complete instructional system's design and development procedures. Typically, as outlined previously in the introduction, they provide tools for implementing the logic and displays specified by the instructional developer for the interactive sequence that he develops. They imply little or nothing in terms of formal instructional strategy support, task analysis, entry population analysis of any of the dozens of other formal procedures which make up a systematic ISD activity. That is, they do not help the CBT author in identifying what the instruction and training problems are, in classifying these problems in terms of the instructional strategies and treatments they imply, or in the rigorous and efficient application of those strategies in terms of reducing the time, cost and variability to generate online instructional and training interactions.

The most that can be said for today's authoring systems is that they tend to say little or nothing about instructional analysis or design while they do provide some support in terms of reducing the computer programming load during CBT development. In addition, some authoring systems do provide some data-collection and analysis tools to aid in the evaluation and revision of CBT instruction. To give a more concrete example, the fully elaborated ISD model for the US Air Force F-16 Pilot Training Course identified 121 detailed procedures and activities in the complete project ISD process. Less than 10 of these were addressed to any significant degree by any authoring system on the market today (O'Neal, Monsees, Carey & Smith, 1977).

What should an authoring system be then? A fully mature authoring system will offer some degree of automated support with online job aiding, helps and instruction for the instructional designer and developer in all phases of the systematic instructional design and development process. For example, it would begin by offering the designer support and tools in the identification of the training problems, including job/task analysis, entry population analysis, training support requirements analysis and so on. Automated tools, although not integrated in any authoring system, do exist for many of these activities. A variety of task analysis database programs are available and have been used with great success in a number of large military training projects in the USA. In the design phase, automated media selection programs have been successfully used in a number of projects and there is no reason why support packages such as the job aids and instructional materials of the DARPA Author Training course could not be implemented in a CBT authoring system environment.

Authoring systems already give considerable help in terms of the development process, shortening times and reducing costs in terms of generating the logic displays, documentation and editing support required for developing CBT materials. Some CMI facilities exist both within and without today's authoring systems, and these offer some level of support to the implementation activities of the project. A variety of data collection and analysis features are also available, which provide some level of analysis and reporting during the formative and summative evaluation phases of the project.

Unfortunately, in many cases, the statistical basis for this capability is rooted in the classical test theory growing out of the empirical instructional development tradition. Many of the statistical models used are of questionable value in the usual CBT environment where one is interested in comparing a student's performance to some absolute standard rather than to the norm-referenced assumptions implicit in comparing students to each other. For example, such standard statistics as item difficulty require some different interpretations in a mastery-learning context. When used on a posttest, what are the implications of items which almost all students get correct? In classical test theory this item would be suspect because it was 'too easy'. In a mastery-learning context, an alternative interpretation might be simply that your instruction was successful and the reason why all students get the item correct is that they have mastered the material.

A powerful family of alternative testing and analysis algorithms is emerging through the computer-adaptive testing (CAT) technology. In this technology, tools are provided for the calibration of items on a variety of dimensions which are then used by 'intelligent' algorithms. Such algorithms select each subsequent item from the item pool for each objective based on the student's performance pattern on previous items on the test. The net result is that a given test reliability can be achieved in approximately 30% of the number of items that a standard test of the same reliability would take. In addition, the adaptive test reliability figure is a much more robust construct which does not break down nearly as quickly when outliers in the population are tested.

A large number of other automated-addressing individual procedures within the ISD process are becoming available. Project planning, spreadsheets and general database management tools are offering powerful and flexible alternatives to early automated ISD attempts such as the author management system (O'Neal & O'Neal, 1979) developed by DARPA. The instructional strategy diagnostic profile (ISDP) developed by Merrill and automated by the Naval Personnel Research and Development Centre as the instructional quality inventory (IQI) provides a powerful tool for the analysis of existing materials in terms of their suitability for addressing individual training objectives (O'Neal, 1977, 1979, 1983, 1984).

History of CBT authoring tools

How did we get to where we are today in terms of authoring systems? The evolution of CBT authoring systems parallels that of other software development tools in computer technology. The first interactive instructional materials developed on computers used the tools of the time, that is general purpose data processing languages. They reflected the advantages and disadvantages of these languages. One advantage of this approach is that the full power of the computer was available to the CBT developer through the flexibility of these languages. However, the obvious disadvantage was that, unless the developer was a sophisticated programmer in his own right, his materials had to be implemented by deputy, that is by someone else who had mastered the arcane arts of programming in that environment. Productivity was typically very low and the expense and time required to code, test and debug the interactive instructional sequences was very high. In addition, the maintenance of curriculum developed in this environment was extremely difficult and expensive. Revision or updating of a course required that one first figure out the idiosyncratic coding style of the original programmer and then, as often as not, the procedure of modifying the course or introducing new materials into the course resulted in the introduction of new bugs into the program with accompanying coding and debugging costs.

For these reasons, specialised syntax in the form of authoring languages specifically designed for CBT emerged relatively early in the history of this technology. By the middle to late 1960s a variety of CBT authoring languages, such as Coursewriter and TUTOR, began to emerge. Typically, these involved specialised syntax for the generation of instructional displays, the eliciting of student responses and answer judging, branching, and scorekeeping. Usually, an author could replace many lines of general purpose programming language code with a few lines of the specialised authoring syntax. Inevitably, however, these languages evolved, as the users generated the need for more functionality, into increasingly complex 'languages' in their own right. As more and more functionality was added to each authoring language it increasingly fell prey to the same productivity and access deficiencies as the general purpose data-processing languages (Fairweather & O'Neal, 1984).

Even at this early time however, there began to emerge structures for the support of genetic instructional strategy implementation. In the Coursewriter II language for the IBM 1500 system, for example, there was an elaborated macro-facility. This allowed an author to define many lines of generic code interacting with variable parameters instead of discrete data. This meant that, at least on a small and restricted level, repetitive patterns within the instruction could be replicated quickly and cheaply without recoding. A more sophisticated example of this trend was the VAULT system implemented in the Coursewriter II environment. This was a more generalised macrofacility that allowed separation of logic and content on a much larger scale. This meant that the CBT developer could effectively articulate a generic instructional strategy with some specificity and the strategy could be replicated with a much lower degree of effort than would be required by use of the authoring language in a more conventional sense.

By the early 1970s, sufficient experience had been gained in the use and elaboration of CBT authoring languages to generate increasing interest in authoring 'system' alternatives. One of the first of these came out of the National Science Foundation TICCIT project. TICCIT was the first system in which the architecture of the authoring process incorporated some very specific (concept and rule-using) instructional strategy considerations into the basic authoring software and student hardware. Because the content being addressed by the project was rich in concept and rule-using objectives, and because a body of research and experience had identified a robust rule/example/practice/help strategy as being very effective in addressing these classes of instructional problem, the authoring software for TICCIT was designed to greatly simplify the process of generating these sorts of instructional components. The control structure for these strategies was a mixture of learner and program control and was implemented at the hardware level through provision of learner-control keys for student use (O'Neal, 1973). A large amount of material in a rich colour and graphics display environment was prepared in a relatively short time (2 years) using largely unskilled labour at BYU. The authoring approach vindicated itself in that it would have been impossible to generate the same volume and quality of material using the same staff in a code-based CBT authoring language. The drawback to the approach, however, was primarily one of flexibility. When the system was used subsequently in a US Navy project, a large number of instructional objectives required other instructional strategies. A more general set of authoring tools, the TICCIT authoring language (TAL), was subsequently added to the basic form driven rule/example/practice authoring format.

The CDS authoring system developed by WICAT in the mid 1970s tackled this problem of productivity versus flexibility in a different manner. In CDS, generic instructional strategies can be developed as menu and form driven interactions for use by the developer. That is, the developer selects the instructional strategy format he wishes to use for a given objective and then responds to a series of menus and prompts which elicit the information necessary to implement the instructional strategy on the content for a given objective. These interactions must be specified by relatively sophisticated instructional designers and are then implemented in terms of author interaction programs (the forms and menus) in a CBT language called CDL (courseware design Language). One major limiting factor in this environment was hardware based. CDS was designed to run on a variety of micro- and small mini-computers. In an attempt to address the transportability problem, it could not be overly optimised for any given machine. Therefore the range of things one could do in CDL was somewhat limited and the relative sophistication and complexity of the instructional strategies which were practical to implement in the system were somewhat limited.

More recent authoring systems such as WICAT's WISE and, within the next l2–18 months, the TICCIT Adapt system, represent what I consider to be a new family of 'hybrid' authoring systems. These systems have several characteristics. Like CDS, they offer the high productivity ease of use and general access to the system afforded by menu-driven prompted, author interaction programs. These systems tend to be easy to learn by nonprogrammers, and productivity within the systems can be extremely high compared to CBT authoring language approaches. At the same time these systems offer access to CBT authoring language sorts of syntax structures to provide the flexibility necessary when one must modify one of the basic 'pre-canned' instructional strategy prototypes or authoring menu alternatives.

In addition, for even more flexibility, these hybrid authoring systems will offer easy access to the general-purpose data processing language environment of the host system(s). For example, in WISE, from any frame of any lesson, the author may provide the student with access to literally anything the computer can run. This includes programs written in Basic, Cobol, C, Pascal Fortran, APL or Assembler, among others. Literally anything the multiuser, multitasking operating system can run on the 32-bit hardware can be provided to any student on any frame of any lesson.

This provides the authoring system with unlimited 'extensibility'. If the author's strategy requires something that is difficult or impossible to accomplish at the menu-driven authoring level, a routine may be written at either the CBT authoring or data processing language level to address the problem. Once this has been done however, such routines are immediately available to other authors, from the menu level of the authoring system. This means, in effect, that you have just extended the range of things which can be accomplished in the authoring system, without coding.

Hybrid systems such as WISE also provide powerful macro and 'prototypic' capability. This means that for any given project, once you have decided upon a basic library of strategies and developed a prototype lesson for each strategy to be used, these generic strategies can be replicated at the macro level with no coding required. The unique content displays, answer processing and feedback for any given objective using one of the strategies can then be added by nonprogrammers from the menu level. Hybrid authoring systems also provide high level 'prototype editors' for easy creation and modification of the basic strategy prototypes.

Status and implications of instructional design theory for today's authoring systems

Figure 1 shows the relationship between cost and increasing productivity in a technology as it matures. Initially the productivity of a given technology will increase fairly rapidly. As the technology matures it takes increasingly amounts of time, effort, money and resources to increase the productivity of the technology by smaller and smaller amounts.

Figure 2 shows the effect of a change of system on a given technology where at certain points, when the cost productivity curve becomes prohibitively flat, innovation in the form of some change of system takes place and for a while productivity increases rapidly again. These sudden inflection points on the productivity versus cost curve are representatives of what Kuhn (1970) would call a paradigm shift, or if you ascribe to

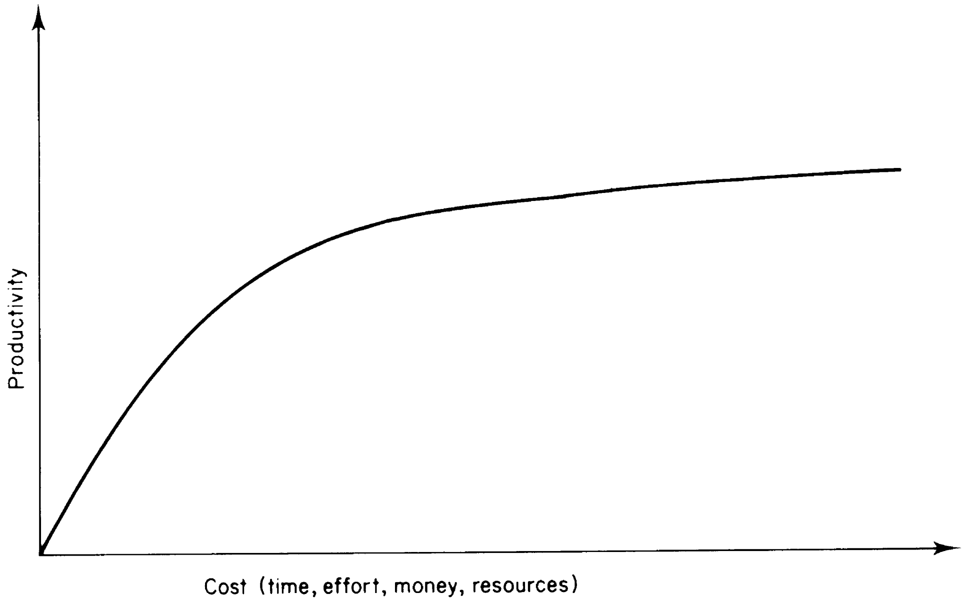

Figure 1: Productivity of a technology

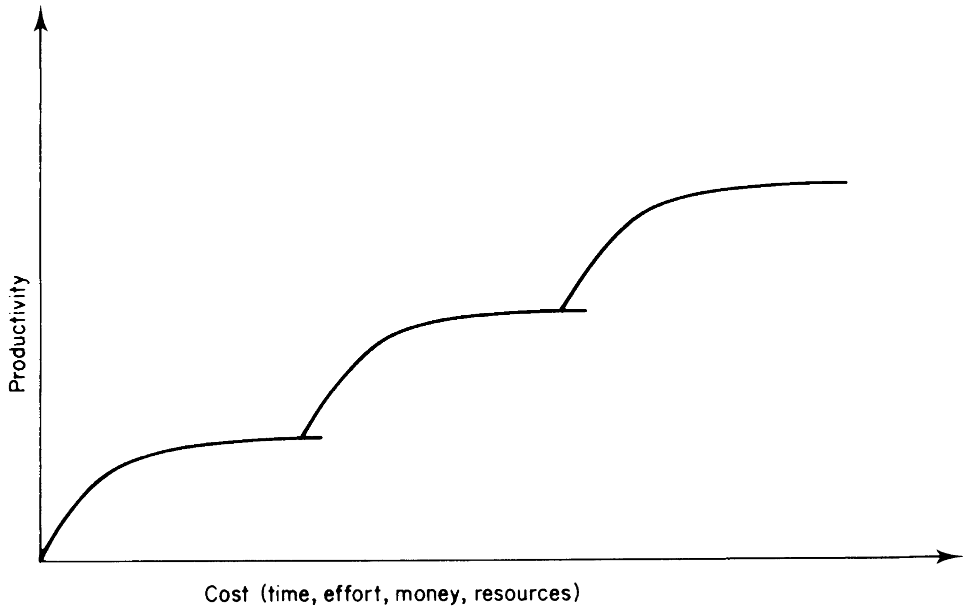

Figure 2: Effect of changing the system in a technology

'catastrophe theory', these represent 'cusps' in the technology development. Figure 3 shows a couple of examples—land transportation and information transmission—where changes in system greatly increased the productivity with respect to the cost of the technology and then flattened out until a new change of system restored the growth curve.

To get a feeling for where instructional design and authoring systems are in this continuing cycle, Figure 4 may be helpful. Notice the position of traditional education and training. The teacher-led instructional model has gone about as far as it can go. We can require teachers and instructors to go to school longer, to take more classes, to pass certification tests, and so on, but until there is a radical change in delivery system, increases in productivity with this approach will be very, very small regardless of how much money is spent.

The systematic instructional design models are not quite as far along yet. We still have many things we can learn and many improvements we can make in instructional design, and continued research and experience will lead to improvements in productivity for some time. However, already gains in productivity through better instructional design and becoming more and more expensive relative to the degree of difference they really make. A quick investigation of the educational and training research literature over the past few years will quickly validate this. Increasingly sophisticated instructional research methodology and models, increasingly sophisticated statistics and

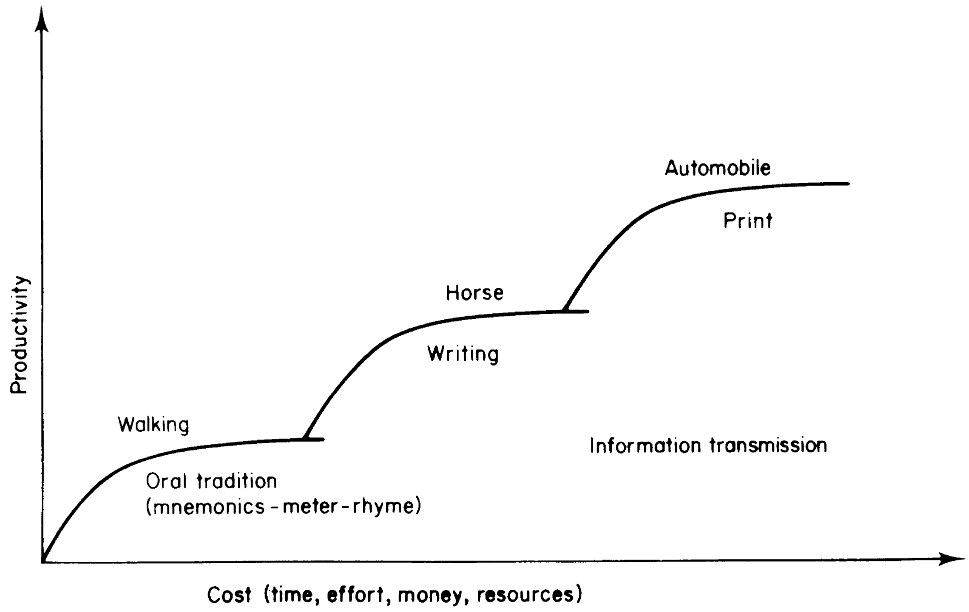

Figure 3: Land transportation

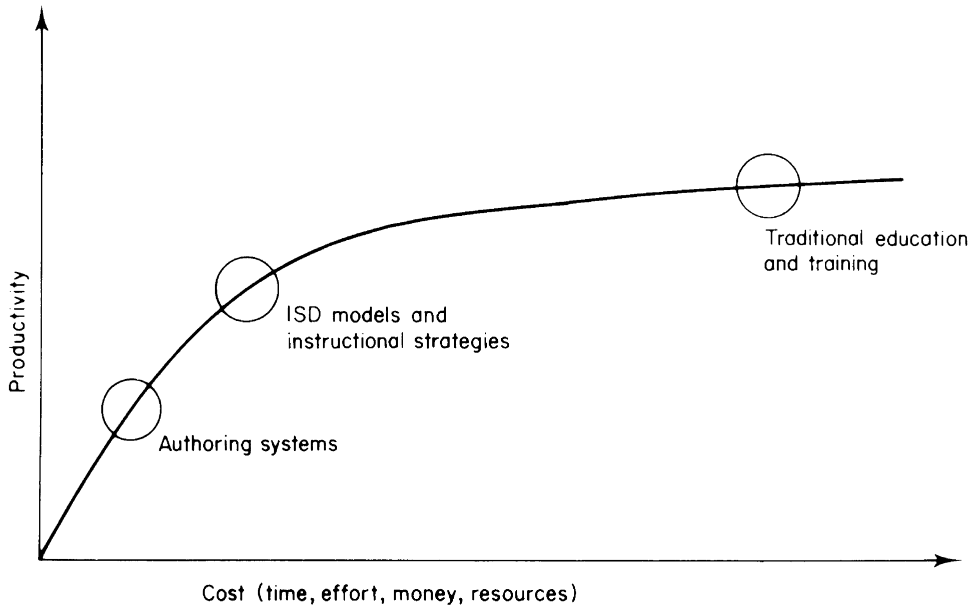
Productivity
Traditional education and training
ISD models and instructional strategies
Authoring systems
Cost (time, effort, money, resources)

Figure 4: Productivity of a technology

widespread application of more powerful computer technology and statistical analysis models are becoming necessary to tease the 'signal' from the 'noise' of instructional research. More simply, this means that we are expending greater and greater effort to identify smaller and smaller changes of effect as a result of our instructional and training research variables. Good symptoms of this decline are such statistical approaches as 'meta-analysis'. In many instructional and training variable research programmes we are operating so close to the limit of improvement that increasing numbers of research studies are showing either no effect or contradictory effects. Meta-analysis has emerged as a necessary tool for teasing out effects, if any, of the instructional variable being investigated across many studies which may individually seem to be contradictory or inclusive. The fact that increasingly in educational research we seem to have to go 'sparrow hunting with statistical elephant guns' is indicative that, perhaps, until there is a radical shift in paradigm (or a cusp) in instructional design theory, we are starting to move into the flat part of the instructional design model productivity curve.

Authoring systems at this point are at a much more immature phase of development in terms of productivity versus cost. We are at the point where great increases in productivity have been, and will be, achieved when compared to the more traditional CBT authoring approaches. Certainly, increasing incorporation of generic libraries of prototypes of today's instructional design models and incorporating a wider range of the total instructional systems design procedures will lead to increases in productivity and effectiveness in CBT authoring. However, the very fact that we can expect an instructional design theory shift, or cusp, in the near future (probably as a result of incorporation of AI techniques in to instructional design) has several implications for authoring systems.

First, authoring systems should be able to create and easily replicate and implement a wide range of flexible and powerful instructional strategies. However, it would probably not be advisable to 'build in' these instructional strategies to the basic architecture of the authoring system at a level which would be difficult to modify or delete in the future.

Second, a much wider range of the ISD procedures and tools must be incorporated into the authoring systems. Again, these should be incorporated at a level of easily accessible authoring system utilities rather than woven through the fabric of the entire authoring system at a basic level. For example, task and job analysis is a venerable and powerful technology in terms of problem identification for instructional design. However, Merrill's elaboration theory and Bunderson's work models offer interesting if not totally compelling alternatives to traditional job/task analysis as a basis for instructional design. Other examples abound. Therefore, in the same sense that an authoring system should be able to implement any instructional strategy and be totally dependent on none, similar flexibility applies to the rest of the ISD model which must be eventually implemented at some level before authoring systems will truly be authoring systems as opposed to merely replacements for coding.

Third, an essential characteristic of authoring systems to allow them to accommodate the full range of utility, functionality and flexibility that is needed before they realise their full potential for increasing productivity, is that any appeal to the CBT authoring language or general-purpose, data-processing-language level of the system need be done only once, and then be fully usable at the menu-driven level in the future by nonprogrammers.

Summary and projections

What lies ahead? Figure 5 shows this writer's estimate of the current status of CBT authoring systems.

I think that since about 1975 we have been in a tool building era where such technologies as systematic instructional design and development have begun to solidify and where a variety of hardware and software tools necessary for real advances in CBT have begun to accumulate. Since about 1982 or 1983 there has begun to emerge an increasing awareness of the need for integrating these isolated utilities and capabilities into increasingly complex systems, validating their effectiveness, and iteratively revising the models to a higher level of utility. Most significant, however, is the third line. We are at the threshold of a technology that will probably contribute the next major system shift both in instructional design theory and in CBT. Artificial intelligence techniques will increasingly affect both instructional design theory and CBT authoring and delivery. Tomorrow's task analysis support program may not be a form-driven database management program or an online task analysis 'lexicon controller' so much as it will probably be an expert systems construct for the identification and classification of instructional and training problems. CBT authoring a decade from now will be less and less of a tool-using activity and more and more of a high-level conversational discourse with an extremely knowledgeable and interesting, but probably non-human, 'colleague'.

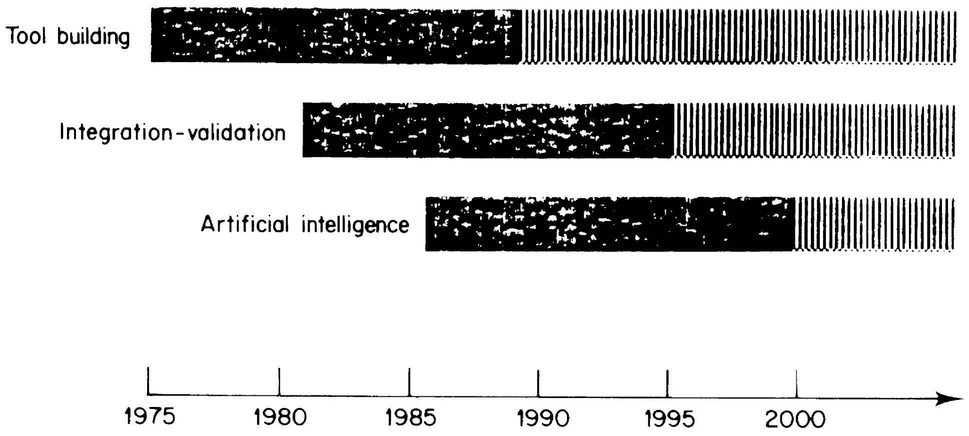

Figure 5: Future authoring systems development

References

Bunderson, C. V. (1973). *Team production of learner-controlled courseware: a progress report*, (Tech Report No. 1). Provo, Utah: Brigham Young University: Institute for Computer Uses in Education.

Bunderson, C. V. (1977). *Analysis of needs and goals for author training and production management systems* (Tech. Rep. No. 1). San Diego, CA.: Courseware Inc. (DOD Contract Number MDA-903-76-C-02 16)

Fairweather, P. G. & O'Neal, A. F. (1984). The impact of advanced authoring systems on CAI productivity. *Journal of Computer Based Instruction*, *11*, 3, 90–94. (Also presented with demonstrations at Man and the Media: International AILA symposium on research tendencies in the teaching of foreign languages with technical aids, Frankfurt, Germany, June 1984).

Kuhn, T. S. (1970). The structure of scientific revolution. In O. Neurath, R. Carnap & C. Morris (Eds), *Foundations of the Unity of Science*, vol II (pp. 53–272). University of Chicago Press.

O'Neal, A. F. (1979, April). *An author management system and other computer based aids to ISD*. Paper presented at AERA, San Francisco. (ERIC No. ED 172 814).

O'Neal, A. F. (1973). *Learner control of instruction: requirements and potentials* (Technical Report No. 2). Provo, Utah: Brigham Young University, Division of Instructional Services, Institute for Computer Uses in Education. (Given as a paper at ADCIS summer session, University of Michigan, 1973) (ERIC Numbers EDO 83805 EMV 11514).

O'Neal, A. F. (1977, April). *Specification and development of computer aids to ISD*. Paper presented at AERA, San Francisco.

O'Neal, A. F. (1983, January). *The evaluation of authoring systems and the automation of ISD*, NSA Conference on E.T. and High Technology, Baltimore, MD.

O'Neal, A. F. (1984, January). *Advanced CBT authoring systems—characteristics and trends*, Technical Application/Language Knowledge Conference (CALICO), Baltimore, MD.

O'Neal, A. F. & O'Neal, H. L. (1979). Author management systems. In H. O'Neal (Ed.), *Issues in instructional systems development*. New York: Academic Press.

O'Neal, A. F., Monsees, J. H., Carey, J. L. & Smith, L. H. (1977). *F-16 training management system needs analysis and design concept* (F-16 Development Report 12). (DOD Contract number FO2604-77-C-0075).

O'Neal, H. L., Faust, G. W. & O'Neal, A. F. (1979). An author training course for resident school environments. In H. O'Neil (Ed.), *Procedures for instructional systems development*. New York: Academic Press.

Pask, G. (1969). Strategy, competence and conversation as determinants of learning. *Programmed Learning and Educational Technology*, *6*, 250–267.