AD-A232 918

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

DTIC FILE COPY

DTIC ELECTE MAR 27 1991 S D D

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION The Regents of the University of California 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Cognitive Science Program Office of Naval Research (Code 1142PT)
6c. ADDRESS (City, State, and ZIP Code) University of California, Los Angeles Office of Contracts and Grants Administration Los Angeles, California 90024	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0395
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22209-2308	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61153N PROJECT NO. RR04206 TASK NO. RR04206-OC WORK UNIT ACCESSION NO. 442c022

11. TITLE (Include Security Classification)

Perspectives on Technology Assessment

12. PERSONAL AUTHOR(S)
Baker, Eva L.; Butler, Frances A., & O'Neil, Harold F., Jr. (eds)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 7/1/90 TO 1/31/91	1991	388

16. SUPPLEMENTARY NOTATION
Papers presented at the Conference on Technology Assessment: Estimating the Future, UCLA, September 5-6, 1990.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Technology assessment
12	05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This is a collection of technical papers based on presentations made at the Conference on Technology Assessment: Estimating the Future. An attempt is being made to secure a commercial publisher for the collection, so circulation is restricted at this time. However, a list of titles follows.

Models and Syntheses

Peled, Z., Peled, E., & Alexander, G. An Ecological Approach for Information Technology Intervention, Evaluation and Software Adoption Policies. Ben Gurion University, Israel.

Clark, R. E. Assessment of Distance Learning Technology. University of Southern California.

(Over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Susan Chipman	22b. TELEPHONE (Include Area Code) (703) 696-4318 22c. OFFICE SYMBOL ONR 1142CS

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

91 3 20 1 3

Models and Syntheses (Continued)

Kulik, J. Assessment of Computer-based Instruction. University of Michigan.

Assessment of Software Strategies

Moore, J. Assessment of Explanation Systems. University of Pittsburgh.

Swigger, K. M. Assessment of Software Engineering. University of North Texas.

Madni, A., & Freedy, A. Concurrent Engineering Technology Assessment. Perceptronics.

Examples of Training and Assessment Technologies

Lesgold, A. Assessment of Intelligent Training Technology. University of Pittsburgh.

Feurzeig, W. Tools for Scientific Visualization. BBN Laboratories.

Goldman, S., Pellegrino, J. W., & Bransford, J. Assessing Programs That Invite Thinking. Vanderbilt University.

Hawkins, J., Collins, A., & Frederiksen, J. Interactive Technologies and the Assessment of Learning. Bank Street College for Children and Technology.

Burns, H. Negotiated Topoi, Networked Epiphanies: Toward Future Technology Assessment Methods and Madness. University of Texas at Austin.

Braun, H. Assessing Technology in Assessment. Educational Testing Service.

Project Report #31

PERSPECTIVES ON TECHNOLOGY ASSESSMENT

Eva L. Baker

Frances A. Butler

Center for Technology Assessment
UCLA Center for the Study of Evaluation

Harry F. O'Neil, Jr.

Cognitive Science Laboratory
University of Southern California

Accesion For		
NTIS CRA&I	√	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

January 1991

Artificial Intelligence Measurement System
Contract Number N00014-86-K-0395

Principal Investigator: Eva L. Baker

Center for Technology Assessment
UCLA Center for the Study of Evaluation

This report was supported by contract number N00014-86-K-0395 from the Defense Advanced Research Projects Agency (DARPA), administered by the Office of Naval Research (ONR), to the UCLA Center for the Study of Evaluation. However, the opinions expressed do not necessarily reflect the positions of DARPA or ONR, and no official endorsement by either organization should be inferred. Reproduction in whole or part is permitted for any purpose of the United States Government.

We wish to thank Judy Miyoshi for her invaluable help at every stage in the preparation of this collection.

Introduction*

This collection of technical papers represents an expansion of ideas presented at the conference "Technology Assessment: Estimating the Future" held at UCLA on September 5 and 6, 1990. The goal for the conference was to identify the major strategies and most promising practices for assessing technology. The papers present perspectives from computer science, cognitive and military psychology, and education. The authors, representing government, business, and university sectors, help to set the boundaries of present technology assessment practices.

The papers are organized into three groups: Models and Syntheses, Assessment of Software Strategies, and Examples of Training and Assessment Technologies. The groups reflect the specific emphases and research interests of the authors within the broad area of technology assessment.

---

*We are currently exploring publication of this collection with a commercial publisher and would thus like to forestall widespread distribution at this time.

Table of Contents

MODELS AND SYNTHESES

AN ECOLOGICAL APPROACH

FOR

INFORMATION TECHNOLOGY INTERVENTION, EVALUATION,

AND SOFTWARE ADOPTION POLICIES

Zimra Peled and Elad Peled
with Gad Alexander

Department of Education

Ben Gurion University of the Negev,

Israel

ACKNOWLEDGEMENT

This article was based on a researech-intervention in Israel [Comptown (Peled, Peled, & Alexander, 1989)] that was generously supported by the Ministry of Education and Culture, Israel, and by the Wolfson Foundation, London. We wish to thank them for their financial and moral support.

We wish to thank the teachers and administrators who participated in project Comptown and to all the staff members of the project who took part in the different phases of the project, without which this study could not be published.

AN ECOLOGICAL APPROACH FOR INFORMATION TECHNOLOGY INTERVENTION, EVALUATION AND SOFTWARE ADOPTION POLICIES

This chapter reviews an approach that guided a five year (1985\86 - 1988\89) research oriented Information Technology [IT] innovation (Project Comptown) that extensively computerized schools and their "close environments" in two localities in Israel (Peled, Peled & Alexander, 1988, 1989). The intervention, evaluation and resulting software adoption policies all evolved from a single conceptual formulation that we call "ecological" (Gibbs,1979). This formulation treats the constituents of large scale interventions and evaluations and combines their multisystemic and treatment-specific components into a model of educational change.

The chapter is divided into three parts. The first part presents the ecological formulation, and briefly describes the Comptown project. The second part elaborates principles, considerations and procedures that are central for evaluating: (a) ecological change processes and (b) treatment-specific experimentation. The third part discusses IT software adoption policy implied by the ecological approach.

The Multisystemic Ecological Formulation

The leading concept in this chapter is presented by a multisystemic ecological formulation based on Bronfenbrenner's theory of nested ecological frameworks (Bronfenbrenner, 1977, 1979) and on ecological notions of educational theories (e.g. Goodlad, 1979; Guba, & Lincoln, 1988; Salomon, 1990; Sarason, 1982). This formulation perceives the ecology of the classroom as a concentric arrangement of four nested systems that act and interact. The innermost and core construct of this arrangement is the classroom, containing learners and teachers. This system consists of three open-ended functional settings (physical, activity, content) in which instruction and learning occur. Next is the school, containing the school administrative staff. This system is the primary operational unit in which resources and policies are transformed into the classroom settings. The third ring comprises the community's political, administrative, business and social systems, containing the community's key personalities as well as the learners' parents. All three systems express needs and expectations, and exercise pressures that may directly affect (advance or disrupt) learning and instruction.

Finally the outer ring includes educational policy making institutions containing elected and appointed officials at the regional and national levels. Through laws, administrative regulations and resource allocation this outer ring, markedly separated from the school and the classroom, may generate new sources of stimulation that either enhance or discourage new developments in the immediate educational system. Figure 1 schematically maps these ecological arrangements.

(Insert Figure 1 Here)

The mapped arrangements are not merely structural. Their mapping is based on five assumptions:

1. Each system consists of implicit cultural, and explicit functional-instrumental components.

2. All the ecological systems are interrelated by a common "cultural blueprint" that sets the pattern for the structures and processes that occur within and across the systems.

3. Classrooms, schools, and social political institutions are culturally dependent systems. To a large extent, their properties and activities are dominated by cultural traditions, and their cultural messages affect behaviors, and have ripple effects in related systems.

4. Ordinarily Cultural functioning is implicit. Its reproduced patterns and processes remain unnoticed. Their effect becomes explicit, and their critical evaluation become possible, only through interventions that introduce enduring innovations into the ongoing activities of the existing systems.

5. Enduring educational innovations are generated and carried on by two types of parallel and mutually stimulating change processes: cultural-ecological and treatment-specific. Cultural ecological processes result from combinations of acting and interacting factors, within and across the ecologically interconnected systems. Treatment-specific (mostly cognitive) processes result from particular treatments that are applied within the ecologically specified educational settings. IT interventions and IT policy decisions must therefore equally aim at the individual participants, the school and classroom, and their expanded ecological environment.

Project Comptown empirically implemented this formulation.

Project Comptown: An Ecologically Oriented IT Intervention

Comptown was designed to create an ecology in which the correspondence between cultural-ecological and individual changes could be identified, one in which the interplay between individual activities and environmental opportunities and constraints (Gallimore, 1990) could be better understood and exploited. The project was therefore carried out in two localities that differed greatly in their demographic, administrative and political characteristics, in their educational agendas, and in their approach to educational issues. (see Table 1).

(Insert Table 1 About Here)

In locality "A" the entire educational system participated in the project. In the second locality (locality "B") only part of the educational system participated. Table 2 provides a summary description of the educational system and the scope of the intervention in each site.

(Insert Table 2 About Here)

The demographic factors, the political and administrative systems, along with other "situational" factors (that emerged throughout the intervention), created contrastable ecological environments in which the project's premises, goals and operational principles were implemented and its posited educational change expectations could be explored and evaluated.

Comptown's Premises, Goals, Operational Principle and Change Perspectives

The project built on three universally applicable and ecologically oriented premises: First, in the "Information Era" the computer is a "major cultural tool" (e.g. Calfee, 1985; Olson, 1985; Papert, 1987; Salomon, 1990; Shavelson & Salomon, 1985) that "defines and redefines man's role in relation to nature" (Bolter, 1984, pp.13). Realizing IT potential in schools augments the educational environment and narrows the gap between the "school culture" (Sarason, 1982) and the "real world culture".

Second, "sound" educational usages of IT (Winkler, Shavelson, Stasz, Robyn & Feibl, 1985), provide opportunities to generate educational innovations and activate unrealized learning and teaching potentials.

Third, collaborative politicians, community leaders and parents create a "supportive ecology" in which "IT culture" (directly or indirectly affecting schools) can germinate.

Following its premises the project's intervention goals were to:

1. Create a computer culture in schools.
2. Utilize the computer's potential for innovative teaching and learning both in and outside school.
3. Create a supportive ecology in which a "computer culture" can expand.

The operational principles (see Appendix 1) complementarily implemented the three goals in each Comptown site. Two of the seven operational principles -- cultivation of supportive attitudes, mobilization of involvement -- emphasized goal "3", aiming at the entire ecology; three of the operational principles -- high density allocation of computers, varied and open computer application, and a system approach -- emphasized goal "1", aiming at the school and its classrooms. Two additional principles -- use of IT-based and non-IT-based instructional strategies in a mindful manner -- emphasized goal "2" and targeted at individual participants.

Comptown introduced these acts to dramatically change the nature of the traditional "Print and Book" dominated classroom. It assumed that the long term intervention would have three additional educational (cognitive) consequences: First, repeated choices mindfully carried on, such as weighing the "benefits" and "costs" of IT and traditional alternatives, will enhance teachers' and learners' awareness of two sets of relations: those prevailing in the "old" setting and its underlying culture, and those prevailing in the "innovative" setting and its underlying culture. Second, these cultural insights will enable learners and teachers: (a) to test their "old" and "new" learning environments by comparison ; and (b) mindfully change their course of learning and teaching as they progressively augment their higher-order thinking skills (Salomon, 1985, 1990; Salomon & Perkins, 1987; Salomon, Perkins & Globerson, in press). Third, understanding the links between IT and older strategies may cultivate two properties that are critical to educational change: (a) an intuitive understanding of the unique contributions of alternative learning environments to the ongoing "cumulative learning process", and (b) the use of multiple perspectives in a learned task.

The implementation of the ecological model in a complex ecological environment followed a multilevel-multisystemic design (described in Peled, Peled & Alexander, 1989; Project Comptown: Intermediate Reports, 1985 - 1987, 1985 - 1989; Project Comptown: IT Treatment Studies, 1987 -1989) that: (a) distinguished preparatory, implementation and adoption of the innovation conditions and functions; and (b) used specified intervention strategies in the classroom and in the nesting systems.

The evaluation of the project showed that unlike the claim often made in "experimental" (e.g. Becker, 1987, 1988; Clark, 1983a, 1983b, 1985a, 1985b; Pea, 1987; Walker, 1987) or "cultural" (e.g. Papert, 1987; Salomon, 1990a, Salomon, 1990b; Scarr, 1985) research literature, processes and outcomes that were demonstrated in IT classrooms were neither specific nor holistic. They rather resulted from two types of interrelated developments: (a) continuous (often long term) and complex ecological developments that were contingent on the specific nesting arrangements of the intervention, and (b) treatment (often short term) generated processes that were realized through interactions between learners and particular IT devices (applied within an ecologically specified framework). The assumptions and concerns that guided this evaluation were not specific to Comptown. They were conceptually rooted in the general ecological formulation.

Major Concerns of an Ecologically Oriented Evaluation

In the ecological formulation the basic structure components are dynamic classroom settings that are conditioned on nested systems. The basic process components are intersystemic and intrasystemic interactions that create and carry on the cultural-ecological and treatment-specific innovations. An evaluation that is guided by these conditioning assumptions is consequently concerned with three issues that are ordinarily bypassed by conventional "input-output" evaluations and are central to the ecological evaluation:

The first issue involves the identification and study of parallel and mutually stimulating cultural-ecological processes that are contingent on the intervention. These processes often act in complex and cyclical ways. Accordingly the evaluation is concerned with: (a) combinations of dynamic factors that contribute to particular results, whereas the relations among these factors and the unique effect of each single factor remain unknown, and (b) developments that need to be studied in cyclical ways, so that new knowledge gained leads to new hypotheses that refer to new and previously unanticipated combinations of factors that both affect and are affected by the intervention.

The second issue is the design of multiple ecological contrasts in which different combinations of structure and process constituents that are not given to experimentation, can be studied and evaluated. This design implies the construction of basic data structures that: (a) formally define the building blocks (facets) of contrasted ecologies, and (b) translate these specifications into reproducible observations.

The third concern of an ecological evaluation is the understanding of the effects of specific treatments that are compatible with the ecological model and are part of the intervention. These understandings which are essential for further intervention manipulations imply the design of an ecologically sensitive experimentation that is theory driven and that rules out counterinterpretations within the well specified ecology.

The following examples from Comptown elaborate these concerns.

Ecological Change Processes: The Comptown Example

In Comptown ecological processes that were repeatedly activated by interactions in each and all ecological systems were realized through interconnected classroom accommodations, school modifications, centralized policies, and beliefs - and attitude - based behaviors. Tables 3, 4, 5 and 6 examine some of the indicative developments in the first three years of the intervention (1985-1988). Each of the four tables focuses on "pre-project" and "project-triggered" characteristics that realize one of the four ecological change processes involved in the innovation.

Table 3 focuses on accommodations (physical, activity and subject-matter content) observed in classroom "6" (school "Y", locality "A"). The developments in this classroom (managed by the same teacher, in the same school and same locality) accurately represent the changing trends in classrooms that actively and continuously tried to implement the project's goals.

(Insert Table 3 Here)

Furthermore, the analysis of the accommodations showed that: (a) the contrasted characteristics realized inseparable aspects of interrelated classroom events; (b) the emerging patterns could neither be understood nor valued in terms of isolated classroom settings, and (c) the observed accommodations were nested: i.e. additional ecological processes that converged in the classroom were differently realized in the "1985" and the "1988" situations.

Table 4 demonstrates some of the school's modifications that framed the classroom's "move" from one educational orientation (frontal teaching) to the other (interactive group work).

(Insert table 4 Here)

The listed administrative, social, and curricular modifications reveal a dynamic school policy that was fruitful both inside and outside the school. Inside the school it reshaped some of the prevailing principles, reinforced existing trends and nurtured interactive relations between the school and its classrooms. Outside the school it produced new ideas that activated local support systems and influenced centralized policy making institutions. The policy modifications of school "Y" and its generated internal and external interactions were typical processes in locality "A". In this locality involvement, support, and intervention actions progressively increased, moving in the same direction. These interdependencies did not develop in locality "B". Table 5 presents examples that demonstrate the different intersystemic interactions in the two Comptown sites (Comptown: Intermediate Reports 1986, 1988).

(Insert Table 5 About Here)

These examples show that the distinctive ecological arrangements of localities "A" and "B" generated different ecological processes. Furthermore, the examples that follow show that these processes interacted differently with the fourth type of ecological change processes: "belief - and attitude - based behaviors" (Jagodzinski & Clarke, 1986).

In Comptown "belief - and attitude - based behaviors" operated across the ecological levels in two ways. First, at the introduction of the innovation, they provided a bridge between the intervention and its unknown results (Schank & Abelson, 1977). Second, as the intervention evolved they tested the initial promises of the intervention against its particular outcomes. Table 6 describes "belief - and - attitude - based behaviors" in the two Comptown sites.

(Insert Table 6 About Here)

Although contradictory in their consequences, "belief - and - attitude - based behaviors played identical roles in the two localities. They aimed at the project's promises and cultivated "theoretical" expectations that were not based on already existing experience. In locality "A", realized expectations augmented positive attitudes toward the project. In locality "B", the real events contradicted expectations, nurtured critical attitude based behaviors, and augmented the negative approach toward the project.

Taken together, the examples presented above reveal: (a) systematic relations between an implicit "community culture" and explicit school and classroom characteristics, and (b) cyclical progressions of dynamic multisystemic ecologies that cannot be modelled by conventional hypotheses testing paradigms

The evaluation of Comptown could not decide, nor could it experimentally find out, whether the radically different project histories reviewed in this chapter resulted from different political and social constellations, different physical settings, different school cultures, different attitude based behaviors, or a combination of each and all of these factors. This experience implies that the evaluation of complex ecological arrangements should build upon a paradigm that models multisystemic linear and non linear functioning and that emphasizes: (a) an explorative, hypotheses generating approach that contrasts ecologies (rather than an hypothesis testing approach that builds upon randomized experiments) and (b) a data structure that formally and empirically characterizes variations within the contrasted ecologies.

Formalization of Basic Ecological Data Structures

The nested systems formulation assumes that classroom occurrences reflect, and permit the tracing of developments across the ecological levels. In ecological evaluations the formal specification of basic data structures is based on this assumption.

Furthermore, the proposed ecological paradigm treats within Guttman's facet theory (Canter, 1985; Guttman, 1957; Shye, 1978) structure and process components that: (a) represent the classroom and delimit its ecology, and (b) translate ecological variations into reproducible observations that can be used in a contrast based evaluation.

To achieve this type of data representation a two stage design is required. The first stage involves the specification of three structure and four process components that are essential and necessary for the design of the ecological evaluation. The second stage elaborates specific properties and processes that characterize each specific phase of the intervention.

The structure components that are essential in a first stage design are the three classroom settings, often discussed in ecological and educational literature. These settings are: <u>"Physical"</u> (e.g. Bronfenbrenner, 1977, 1979; Calfee and Brown, 1979; Project Comptown, 1988, 1989, 1990); <u>"Activity"</u> (e.g. Doyle, 1986; Lamm, 1976; Leontiev, 1964; Project Comptown, 1988, 1989, 1990; Vygotsky, 1978), and <u>"Content"</u> (e.g. Doyle, 1988; Lamm, 1976; Project Comptown, 1988, 1989, 1990; Shavelson, Winkler, Stasz, Robyn & Shaha, 1984).

The process components that are essential in a first stage design are four facets that enable characterization and contrast of the "newly introduced" cultural-educational frames, and the pre-project "old" frames. These process facets are: (a) the inventory of the items that distinguish the setting; (b) the organization of the items within the setting; (c) the intrasystemic accommodations that involve the setting, and (d) the intersystemic relations that affect the settings.

Since any situation or event in the classroom can be characterized as a particular combination of these settings and facets, all three settings and four facets are considered essential and necessary for the design of the evaluation. Table 7 presents preliminary, first stage, specifications for the construction of reproducible ecological observations.

(Insert Table 7 Here)

Tables 3, 4, 5 and 6 can be taken as typical observations that built upon these specifications. The presented examples were derived from Comptown's longitudinal exploration of multiple ecologies that contrasted: same classroom within the same school, different classrooms within the same school; classrooms in different schools within the same locality; different schools within the same locality, and schools in different localities. Figure 2 schematically lays out the ecological units of analysis used in this study.

(Insert Figure 2 Here)

The Comptown intervention was carried out in three phases: (a) preparation, (b) implementation, (c) adoption of the innovation. Each phase had different foci and different characteristics that first stage design could not capture. The second stage design introduced detailed refinements that clarified specific objectives and referent systems and thereby operationalized first stage specifications.

The ecological paradigm developed in this section has four distinguished characteristics:

1. Properties and processes that have no effect on classroom occurrences, and that cannot be traced through properties and processes in classroom settings, are omitted from the evaluation design.

2. The units of analysis are the natural systems. Behaviors that are included in the design are not separated from their natural settings and can be studied as culturally and ecologically dependent behaviors.

3. The settings and facets that guide the construction of the observations remain intact across different ecological arrangements.

4. The two stage design allows for additions, corrections, deletions, and accommodations that conceptually elaborate and empirically validate the ecological model.

Ecologically Sensitive Experimentation

The ecological assumption implies that IT treatments cannot be separated from their ecological contexts, or from the cultural comparisons and educational activities they enhance. Accordingly IT treatments should be defined in terms of (a) interactions between learners in an ecologically characterized classroom, and a particular type of educational software, and (b) the alternating media representations introduced. The "ecologically sensitive experimentation" that we propose elaborates this formulation in four facets. These facets are: (a) "Type of Information Exchanged and Processed" in the interaction (i.e. subject matter bound or procedural and applicable to a variety of subject matters (Give'on, 1988)); (b) "Cognitive Demands Set on the Learner" (i.e., low-reactive or high-level-interactive processing demands (Give'on, 1988; Salomon, 1985; Salomon & Globerson, 1987; Salomon & Perkins, 1987)); (c) "Proximity between Processes" generated throughout the interaction and processes that are encouraged by the already functioning educational frameworks (i.e., processes producing "more of the same" and leaving the learner's "zone of proximal development" (Vygotsky, 1978; Wertsch, 1985a, 1985b) intact, or processes that trigger an innovative experience that augments the learner's "zone of proximal development" (Project Comptown, 1990)); (d) "Exposure to Alternating Representations" (i.e., use of software in a non selective or a critical selective manner (Gal, 1990; Bamberger & Schon, 1983). Table 8 presents faceted design for the characterization of IT software used in ecologically sensitive experimentations.

(Insert Table 8 About Here)

Since IT treatments are nested within natural ecological settings, the generality and validity of the treatment cannot be technically handled. They must be defended in two ways. First, the treatment and the stimulated cognitive processes must be relevant to a variety of learning and teaching contexts. Second the implementation of the treatment and control conditions must be equivalent and representative in relation to similar learning and instruction activities in regular classroom settings. A design that satisfies these conditions, and tests the implications of a mindful use of a particular type of software in natural classroom settings, is presented in Table 9.

(Insert Table 9 Here)

This design was repeatedly used in Comptown for the evaluation of the cognitive benefits of a mindful use of types of software that encourage the development of higher order cognitive skills and meet the ecological criteria of generality and validity[1]. The "Algebraic Linear Functions" and the "Data Base" studies are two examples. The "Algebraic Linear Functions" study evaluated the relation between a mindful use of "computer generated" simulations and the users' ability to abstract the basic principles of these functions and apply these principles to new types of algebraic functions. This study was carried out over six months (teacher training included) with eighth and ninth grade students and was concluded with three tests: a regular achievement test that evaluated learners' ability to apply familiar algebraic principles to familiar problems; a "one step" inference test that evaluated learners' ability to apply familiar principles to new and unfamiliar problems, and a "multiple inference" test that evaluated learners' ability to use general abstractions for the generation of new principles and to apply the newly generated principles to new and unfamiliar problems. Interestingly, classrooms that selectively and critically used computer simulations, coped best with "multiple inference" problems.

The "Data Base" study evaluated the relation between the mindful use of data-base software and the learners' ability to organize data, identify meaningful variables, and formulate and test hypotheses. This study was carried out over six months (teacher training included) with sixth-grade students and was concluded with a "paper and pencil" test that evaluated learners' ability to apply "data-base" inquiry procedures to new information that is not studied in schools. The results indicated that a mindful use of computer "Data-Base" software might help to realize IQ potentials. The correlations between IQ and "data-base" scores in the "treatment" classrooms were consistently and significantly higher as compared to the same correlations in the "control" classrooms.

Comptown's ecologically sensitive experimentation was carried out in natural classroom settings that were "ecologically controlled". Using methods that are ordinarily associated with quasi-experimental studies (Campbell & Stanley, 1963), the ecologically sensitive experimentation tested specific causal hypotheses that were theory driven. However, this does not imply that the theories from which the hypotheses were derived suggest that the highly complex multisystemic ecological factors do not affect or are not affected by the intervention. They rather emanated from the need to know which aspects of the complex classroom ecology deserve more attention. Comptown's experimentation therefore carefully looked at the ways in which the systemic conditions in the treatment classrooms differed from their controls. One paramount characteristic was the congruence between the generic properties of the instruments of learning and teaching and the teacher's pattern of instruction. This finding motivated the construction of a "Taxonomy for Information Technology Adoption Policy" (Peled, Peled, & Alexander, to be published): an IT software selection procedure that is implied by the ecological approach.

A "Taxonomy for Information Technology Adoption Policy: An IT software Selection Procedure Implied by the Ecological Approach

In the ecological model a pattern of instruction and the use of instruments of instruction derive from a single conceptual framework that to a large extent is culturally-ecologically dominated. An intervention that introduces new instructional instruments may either disrupt the "Instruction-Instrument" harmony and reveal the multifaceted nature of these implicit, ordinarily "hidden" connections, or lead to inadequate and fruitless use of the newly introduced technologies. Recognizing the vital importance of adequately matching teacher and instrument, Comptown's ecologically sensitive experimentation introduced IT software while changing the teacher's pattern of instruction. The "Taxonomy for Information Technology Adoption Policy" tries to generalize the Comptown experience and reveals the possible correspondences between the generic properties of patterns of instruction and six types of IT software ordinarily used in education (Drill & Practice, Tutorials, Games, Simulation, Open Tools, Educational Programming).

More specifically, the "Taxonomy for Information Technology Adoption Policy" consists of three components: one that characterizes instructional processes which determine teachers' patterns of instruction; one that specifies properties of software, and one that defines the range of congruence between instructional processes and software.

Instructional processes are further elaborated by five decision making processes (facets) that delimit the diversity of patterns of instruction. Software properties are further elaborated by five facets that specify these properties in terms of their managerial and instructional qualities (i.e., elements of these facets reinforce elements of practiced patterns of instruction). Table 10a presents the five instructional decision making facets and their elements.

(Insert Table 10a About Here)

Table 10b presents the five software facets and classifies educational software accordingly.

(Insert Table 10 b About Here)

The "Linear Functions" and "Data Base" experiments are concrete examples that demonstrate the benefits of congruence. In these experiments congruent software properties augmented instructional methods. We will use these studies as representative applications of the more general framework delimited by the "Taxonomy".

The instructional goals emphasized by the linear function "treatment" teachers (who internalized the project's operational principles and were trained to use the simulation software mindfully) were: (a) derivation of a set of abstract principles (Facet [A]: element 3) that permit each student to rediscover abstract sets of relations and construct new functions (Facet [D]: element 3) and (b) development of computer skills that allow each student systematically and mindfully manipulate computerized simulations (Facet [A]: elements 1, 2). The learning activities enhanced by these teachers were either group directed or individualized and autonomous (Facet [C]: elements 2, 3). To this end the treatment - classroom teachers were constantly engaged in curricular decisions (Facet [B]: element 2) that involved: (a) mapping of elements of instructional patterns onto elements of the simulation software and (b) selection and integration of computer-based activities into the on going instruction and learning procedures.

The software's managerial properties enabled teachers to deliver some "control" responsibilities (Facet [F]: element 2) to software users and act as facilitators (Facet [E]: element 2). The software's information processing properties reinforced the teachers' tendency to enhance: (a) use of semi structured learning materials (Facet [I]: element 2), (b) joint computer and user generation of information (Facet [H]: element 2), and (c) discovery of software's algorithm (Facet [J]: element 2).

Similar developments took place in the "Data Base" study. The "treatment" teachers emphasized the acquisition of computerized data base strategies and higher order information handling skills (Facet [A]: element 4). Responsibility for the design of the content, to a large extent, devolved on software users (Facet [B]: elements 2, 3). Teachers facilitated (Facet [E]: element 2) group directed (Facet [c]: element 2) learning activities that were essential to new knowledge construction operations (Facet [D]: elements 2, 3).

The "Data Base" software information processing requirements induced learners to supply information and use software's algorithm for the construction of new knowledge (Facet [H]: element 3; Facet [J]: element 3). These requirements intensified an already existing pattern of instruction developed by teachers who had internalized Comptown's operational principles. Hence, data construction procedures could be practiced in an enriched educational environment.

Both experiments spell out congruence in terms of fluid interdependencies between instructional patterns and instructional properties of software and demonstrate pedagogical, psychological and administrative implications of software selection and use. Furthermore, the experiments prove that congruence rests on a dual verification process: examination of the elements that characterize a pattern of instruction and examination of the correspondence between a pattern of instruction and generic software properties. Table 11 maps elements of patterns of instruction on elements of software properties and reveals optional software and instructional cross classifications that may lead to educatioal change.

(Insert Table 11 About Here)

Concluding Remarks

This paper reviews an ecological formulation that applies equally to information technology intervention, evaluation and software adoption policies. The formulation is based on the assumptions that: (a) classrooms, schools, and social political institutions are culturally dependent systems, (b) their cultural functioning, ordinarily implicit, becomes explicit through interventions that introduce innovations into the on-going activities of the existing systems, and (c) enduring educational innovations are generated and carried on by two types of parallel and mutually stimulating change processes: cultural-ecological and treatment-specific processes.

Furthermore, the elaboration of this overarching formulation leads to the complementarity of three paradigms: a paradigm that models processes evolving from complex cultural-arrangements that cannot be subjected to experimental manipulations: a paradigm that models processes evolving from treatment manipulations in quasi-experimental settings, and a paradigm that models decision making considerations that are implied by the intervention and may yield congruence or dissonance in the immediate educational environment.

These paradigms prescribe the use of fundamentally different methods within the same intervention and same evaluation. The cultural-ecological framework leads to the design of multilevel systems intervention that requires longitudinal explorations of multilevel systems with ever changing interdependencies (linear and non linear). The treatment-specific intervention leads to the introduction and manipulation of "external", independent variables. The evaluation of these treatments imply quasi-experimental framework that focus on causal relations with comparison of equivalent groups, and analyses of sequences of discrete events that offer predicted and measurable results, in response to narrow questions. The congruence framework combines cultural-ecologicl and quasi-experimental methods that facilitate analyses of educational innovations in progress.

The facet definitions of the evaluation framework directly evolve from a project's operational principles of the type under discussion in this chapter. These definitions: (a) guide the design of reproducible observations for the cultural-eclogical, treatment-specific and congruence studies, and (b) generate the entire map of multisystemic functioning that create the educational innovation.

The ecological approach has four additional characteristics:

1. It uses the same conceptual formulation for both the design, and the evaluation of the intervention, eliminating inconsistencies between intervention and evaluation.

2. It accords importance to ecological contexts and if necessary, trades off internal for external validity.

3. It examines the assumption that multiple ecological contexts and multiple educational treatments encourage the development of higher order skills, while single-context educational environments are less likely to do so.

4. It tries to maintain mutually stimulating and constructive relationship between parallel hypotheses-generating and hypotheses-testing operations.

Finally the ecological approach suggests five indicators that are useful for the identification of ecological change processes important for education.

1. Realization of parallel and mutually stimulating change processes: cultural-ecological" and treatment specific.

2. Extensive and varied interactions with IT devices that are mindfully integrated into the "Print and Book" dominated classrooms.

3. Use of multiple perspectives in learning and teaching

4. Awareness of the unique contributions of alternative and alternating learning environments

5. Mindful developments of new and changing courses of learning and teaching that result from repeated comparisons of the "Print and Book" and the "IT" learning environments.

REFERENCES

Bamberger, J. and Schon, D. A., (1983). Learning as reflective conversation with materials: Notes from work in progress, Art Education, 56, (2), 68-73.

Becker, H. J.(1987). The impact of computer use on children's learning: What research has shown and what it has not. (Report No. 18). Baltimore, Maryland: The Johns Hopkins University. Center for Research on Elementary and Middle Schools.

Becker, H.J. (1988). A Field Experiment at a Distance: Studying on a National Scale the Effectiveness of Instructional Practices that Use Computers. Paper presented at the annual meeting of the American Educational Research association, New Orleans, AprilBecker 1988

Bloom, Benjamin S., Engelhart, Max D., Furst, Edward J., Hill, Walker H., & Krathwohl, David R. (eds.). (1956). Taxonomy of Educational Objectives. New York: David McKay Company, Inc.

Bolter, D. J. (1984). Turing's man: Western culture in the computer age. Chapel Hill, NC: The University of North Carolina Press.

Bronfenbrenner, U. (1977) Toward an experimental ecology of human development. American Psychologist 32: 513-531.

Bronfenbrenner, U. (1979). The ecology of human development. Experiments and design. Cambridge, Mass.: Harvard University Press.

Calfee, R. (1985). Computer Literacy and Book Literacy: Parallels and Contrasts. Educational Researcher, Vol. 14, No. 5, May. pp. 8-13.

Calfee, R. & Brown, R. (1979). Grouping Students for Instruction. In Duke, D. L. (1979). Classroom Management: The Seventy-eighth Yearbook of the National Society for the Study of Education. Chicago: NSSE. pp. 144-181.

Campbell, D. T. & Stanley, J. C. (1963). Experimental and Quasi-Experimental Designs for Research . Skokie, Ill.: Rand McNally.

Canter, D. (Ed.) (1985). Facet Theory: Approaches to Social Research. New York, Berlin: Springer-Verlag.

Clark, R. E. (1983a). Reconsidering Research on Learning from Media. Review of Educational Ressearch, 53, pp. 445-459.

Clark, R. E. (1983b). Evidence for Confounding in Computer-Based Instruction Studies: Analyzing the Meta-Analyses. Educational Communication and Technology Journal, Vol. 33, No. 4, pp. 249-262.

Clark, R. E. (1985a). Confounding in Educational Computing Research. Journal of Educational Computing Research, 1(2), pp. 137-148.

Clark, R. E. (1985b). The Importance of Treatment Explication: A Reply to J. Kulik, C-L. Kulik and R. Bangert-Drowns. Journal of Educational Computing Research, Vol. 1 (4).

Clements, Douglas H. (1989) Computers in elementary mathematics education. glewood Cliffs, New Jersey: Prentice Hall.

Cohen, D. K. (1988). Educational technology and school organization. In Nickerson, R. S., & Zodhiates, P. P. (eds.) Technology in Education: Looking Towards 2000. Hillsdale, N.J. Lawrence Erlbaum Associates, Publishers.

Doyle, W. (1986). Classroom Organization and Management. In M. C. Wittrock (ed.) Handbook of Research on Teaching. Third edition. New york: MacMillan Publishing Company. pp. 392-431.

Doyle, W. (1988). Curriculum in teacher education. Vice Presidential address presented at the annual meeting of the Amerivan Research Association, New Orlcans, April 5-9, 1988.

Gal, S. (March, 1990). Knowledge-Based systems, and more: A multimedia environment for learning structural engineering. Paper presented at American Association for Artificial Intelligence, Spring Symposium, Stanford university, CA, USA.

Gallimore, R. (1990), Rousing School Reforms to Life. Prepared for an Invited Address to the Annual Meeting of the Social Contexts of Education Division, American Educational Research Association, Boston, April, 1990.

Gibbs, J. C. (1979). The Meaning of Ecologically Oriented Inquiry in Contemporary Psychology. American Psychologist, Vol. 34, No. 2. February, pp. 127-140.

Give'on, Y. S. (1988). Taxonomy of educational tsoftware. Proceedings of The Fifth International Conference on Technology and education (480-483). Edinburgh.

Goodlad, J. L. (1979). Perspectives on Theory, Research, and Practice. In Duke, Daniel L. (1979). Classroom Management: The Seventy-eighth Yearbook of the National Society for the Study of Education. Chicago: NSSE. pp. 391-412.

Guba,E. & Lincoln, Y. S. (1988). Do inquiry paradigms imply inquiry methodologies? In D. M. Fetterman (Ed.) Qualitative approaches to evaluation in education: The silent scientific revolution. New york, Praeger.

Guttman, L. (1957). Introduction to Facet Design and Analysis. In Proceedings of the Fifteenth International Congress of Psychology, Brussels. Amsterdam: North-Holland Publishing.

Henson, K. T. & Borthjwick, P. (1984). Matching Styles: A Historical Look. Theory and Practice. Vol. 23. No. 1, pp. 3-9.

Jagodzinski, A. P . Clarke, D. D. (1986). A review of methods for measuring .nd describing users' attitudes as an essential constiteunt of systems analysis and design. The Computer Journal, Vol. 29, NO. 2. pp. 97-102.

Lamm, Z. (1976). Conflicting Theories of Instruction: Conceptual Dimentions. Berkeley, Ca.: McCutshan Publishing Corporation.

Leontiev, A. N. (1964). Problems of mental development. Washington, D.C.: U.S. Joint Publication Research service.

Olson, D. R. (1985). Computers As Tools of the Intellect. Educational Researcher, Vol. 14, No. 5, May. pp. 5-8.

Papert, S. (1987). Computer Criticism vs. Technocentric Thinking. Educational Researcher, 16(1), June-July. pp. 22-30.

Papert, S. (1990). Introduction. In E. Harel (Ed.). Constructionist Learning. Cambridge, Mass. The Media Laboratory, MIT.

Pea, R. D. (1987). The Aims of Software Criticism: Reply to Professor Papert. Educational Researcher, 16 (1), June-July. pp. 4-8.

Peled, E., Peled, Z., & Alexander, G. (to be published). A Taxonomy for Information Technology Adoption Policy.

Peled, E., Peled, Z., and Alexander, G. (1989a). New Information Technology in Education: A Challenge to Educational Policy in Israel. Journal of Research on Computers in Education. Vol 22, No. 1, Fall.

Peled, E., Peled, Z., and Alexander, G. (1989b). Project Comptown: educational intervention and action research. British Journal of Educational technology. Vol. 20. Number 2. May.

Perkins, D. N. (1985). The fingertip effect: How information processing teachnology shapes thinking. Educational Reasercher, 14, 11-17.

Project Comptown (1986). Ingtermediate Report 1985 - 1986. Beer Sheva, Israel: Ben Gurion University, Dept. of Education. Project Comptown (Hebrew).

Project Comptown (1988). Ingtermediate Report 1985 - 1987. Beer Sheva, Israel: Ben Gurion University, Dept. of Education. Project Comptown (Hebrew).

Project Comptown (1989). An intermediate report: IT treatment studies 1987-1989). Beer Sheva, Israel: Ben Gurion University, Dept. of Education. Project Comptown (Hebrew).

Project Comptown (1990). An Observational Report. Beer Sheva, Israel: Ben Gurion University, Dept. of Education. Project Comptown (Hebrew).

Salomon, G. (1985). The new information technologies: What you see is not (always) what you get. Educational psychologist, 20, pp.207-217.

Salomon, G. (1990). Transcending the qualitative/quantitave debate: The analytic and systemic approaches to educational research. (Unpublished paper).

Salomon, G. (In Press). Studying the flute and the orchestra: Controlled vs. classroom research on computers.

Salomon, G., & Globerson, T. (1987). Skill may be not Enough: The Role of Mindfulness in Learning and Transfer. International Journal of Educational Research, 11, pp. 623-637.

Salomon, G., & Perkins, D.N. (1987). Rocky Roads to Transfer - Rethinking Mechanisms of a Neglected Phenomenon. Paper Presented at the Harvard University Conference on Thinking.

Salomon, G. Perkins, . & Globerson, T. (in Press). Patterns in cognition: Extending human intelligence with intelligent technologies.

Sarason, S. B. (1982) The Culture of School and the Problem of Change. Second Edition. Boston: Allyn and Bacon, Inc.

Scarr, S. (1985). Constructing psychology: Making facts and fables for our times. American Psychologist, 40, pp. 499-512.

Schank, R., & Abelson, R. (1977). Scripts, Plans, Goals and Understanding. New York: Wiley.

Shavelson, R. J. & Salomon, G. (1985). Information technology: Tool and Teacher of the Mind. Educational Researcher, Vol. 14, No. 5, May. p. 4.

Shavelson, R,J., Winkler, J. D., Stasz, C., Feibel, W., Robyn, A. E., & Shaha, S. (1984). Teaching Mathematics and Science: Patterns of Microcomputer Use. Santa Monica: Rand.

Shye, S. (1978). On the Search for Laws in the Behavioral Sciences. In Shye, S. (ed.), Theory Construction and Data analysis in the Behavioral Sciences. San Francisco: Jossey - Bass Publishers, pp. 2-24.

Vygotsky, L. S. (1978) Mind in Society. Cambridge, Mass.: Harvard University Press.

Walker, D. F. (1987). Logo Needs Research: A Response to Papert's Paper. Educational Researcher, 16 (1), June-July. pp. 9-11.

Wertsch, J. V. (1985a). Vygotsky and the Social Formation of Mind. Cambridge, Mass.: Harvard University Press.

Wertsch, J. V. (Ed.). (1985b). Culture communication, and cognition: Vygotskian perspectives. Cambridge: Cambridge University Press.

Winkler, J. D., Shavelson, R. J., Stasz, C., Robyn, A. E. & Feibel, W. (1985). Pedagogically sound use of microcomputers in classroom instruction. Journal of Educational Computing Research. 13, pp. 285-293.

NOTES

1. The design of each of the two experiments combined:

[a] A careful selection of an IT treatment that involved formal thinking operations (Piaget & Inhelder, 1969), and was either applicable to different levels of abstraction within a specific content area (algebraic functions), or to a variety of subject matters.

[b] An intervention strategy that allowed the research to use reproducible methods in the characterization the manipulation of the software throughout the different steps of the intervention.

[c] The definition and the prediction of measurable outcomes

[d] The use of control an treatment group with equivalent student distributions, equivalent classroom settings, and equivalent nested systems in variety of subject matters

Table 1: Demographic, Administrative & Political Characteristics of the Two Comptown Sites

CHARACTERISTICS	LOCALITY "A"	LOCALITY "B"
DEMOGRAPHIC		
Population	13 000	54 000
Composition	homogeneous mainly middle class	heterogeneous
ADMINISTRATIVE & POLITICAL		
Educational administration	centralized	decentralized
Interest groups	insignificant	pronounced
Political pressure	insignificant	dominant
Main educational issues	improvement & innovations of schools	provision of quality education to large groups of advantaged & disadvantaged students

Table 2: A Summary Description of the Educational Systems and the Scope of the Intervention in the Two Comptown Sites

CHARACTERISTICS	LOCALITY "A"	LOCALITY "B"
GENERAL		
Scope of the experiment	the entire city	three neighborhoods
School structure		
elementary school	grades 1-6	grades 1-8
secondary school	jun. high 7-9 sen. high 10-12	grades 9-12
SCOPE OF THE INTERVENTION		
Elementary schools		
Schools	6 (all)	9 (of 23)
Students	2, 000	3, 400
Teachers	200	260
Secondary Schools		
Schools	1	none
Students	1, 430	none
Teachers	125	none

Table 3: Indicative Physical, Activity and Content Accommodations in Classroom "6", School "Y", Locality "A" (1985, 1988)

ACCOMMODATION	1985	1988
PHYSICAL: Instructional aids	"Whole Class" oriented	"Group Work" oriented
Table & chair organization	Facing classroom's front (facilitating frontal teaching)	Optionally organized in activity centers for group work
ACTIVITY: Essential student learning activities	Attentive listening & imitating	Group & machine interactions; Exploring & modelling
Classroom behavior	Essentially controlled & quiet; Occasional misbehavior	Noisy and constantly moving; Minimal misbehavior
Dominant teacher delivery modes	Lecturing & demonstrating	Tutoring & guiding group & student machine interactions; Using computer based activity as a main delivery channel
CONTENT: Curricular goals	Knowledge & basic skills acquisition	Development of new & enriched content areas such as: handling of information; use of multiple technologies & multiple knowledge sources

Table 4: Administrative, Social and Curricular Modifications in School "Y", Locality "A" (1985, 1988)

POLICY MODIFICATION	1985	1988
ADMINISTRATIVE: Allocation policy of instructional aids	Supply of identical aids to different classrooms	Supply of aids chosen by the teacher to meet her\his needs
SOCIAL: Role assignment	Complete separation between teaching & learning activities & responsibilities	Assignment of technical & teaching roles to students (e.g. older students teach the young)
CURRICULAR: Curricular emphases	Basic skills acquisition & accumulation of information	Development of interdisciplinary inquiries; Information handling & use of multiple learning & teaching strategies
Special programs	Special extra classes for low achievers;	Promotion of IT based school activities: e.g. computer game library; school desk top publishing; IT training for teachers in & outside school; Parents IT training

Table 5: Community Support and Centralized Educational Policies in the Two Comptown Sites (1985, 1988)

CENTRALIZED POLICY	LOCALITY "A"	LOCALITY "B"
PHYSICAL SUPPORT: Infrastructure & equipment	1985: Firm municipal commitment to build an IT infrastructure; Firm commitment of the Ministry of Education to support introduction of IT infrastructure & supply computers to schools	1985: Vague municipal commitment to build an IT infrastructure; Vague commitment of the Ministry of Education to support introduction of IT infrastructure & supply computers to schools
	1988: Private contributions of computers to schools; Municipal maintenance of IT infrastructure & of computers installed & used in classrooms	1988:
PROFESSIONAL SUPPORT: Planning	1985: Municipal appointment of experts to advise purchase of computers & develop networking	1985: Municipal appointment of experts to advise purchase of computers & develop networking
	1988: Experts follow up development of networking	1988:
Special programs:	Promotion of special training programs (e.g. parents & senior citizens training)	
SOCIAL SUPPORT: Involvement	1985: Voluntary parents activity in classrooms	1985: parents attempt to affect intervention programs
	1988: parents' participation in the project's inside & outside school activity	1988:

Table 6: Belief and Attitude Based Behaviors in the Two Comptown Sites

SYSTEM	BELIEF AND ATTITUDE BASED BEHAVIOR LOCALITY "A"	LOCALITY "B"
MINISTRY 1985:	Positive attitude toward the research project generates support in the project	Lack of confidence in the local authorities generates limited support in the project
1988:	Positive field outcomes & current demands result in longtime commitments	Attitudes & support remain unchanged
LOCAL POLITICAL AUTHORITIES 1985:	Convinced that computer culture in schools serve local interests; participate in progress & difficulties	Approach Comptown as one of the competing projects; make no effort to support
1988:	Demonstrate consistent attitudes; increase involvement & commitment to the project's success	Approach & support remain unchanged
SCHOOL PRINCIPALS 1985:	Led to believe that the promised innovations can improve learning & teaching; set priorities that match the project's operational principles	Led to believe that the innovation can improve learning & teaching; express support in the project
1988:	As the project moved towards promised targets augment involvement & express specific interests	Doubt & dispute the project's policies & disappointed from results
TEACHERS 1985:	Convinced that additional work caused by the project is worthwhile; voluntarily participate in training	Internalize the project's objectives; voluntarily participate in training
1988:	Gained success in work leads to interest & involvement in the achievement of specific goals	Difficulties to implement the project's working principles result in no commitment while continuing to use the training offers of the project

(To be continued)

(Table 6 continued)

SYSTEM	BELIEF AND ATTITUDE BASED BEHAVIOR LOCALITY "A"	LOCALITY "B"
PARENTS 1985:	Believe the computers carry a promise for better education; They support & are involved in the project	
1988:	Responding to enthusiastic reactions of children parents involvement is increased	

Table 7: Preliminary First Stage Specifications
for the Construction of Reproducible Ecological Observations*

FACET SETTING	INVENTORY	ORGANIZATION	INTRASYSTEMIC ACCOMMODATIONS	INTERSYSTEMIC CHANGES
PHYSICAL	Available: * Technology * Materials * Learning aids	Supporting: * Whole class * Group * Individual learning & instruction	Accommodations in physical inventory and\ or organization appear in combination with accommodations in additional settings	Demonstrated changes in: * Social * Political * Other * Involvement * Support * Commitment
ACTIVITY	Enacted: * Management style * Delivery mode * Learning mode Misbehaviors * Minimal * Occasional * Severe	Supporting: * whole class learning * Group interactions * Individual learning * Student x machine interaction Session: * Shortened * Regular * Shortened	Changes in activity inventory and\ or organization appear in combination with accommodation in additional settings	Demonstrated changes in: * Social * Political * Other * Involvement * Support * Commitment * Other
CONTENT	Enacted: * Goals: Social; Instruction: Technical Curricular emphases: * Basic Skills * High order skills * Technical skills * Other Enhanced Motivation: * high * Average * Low	According to: * Hierarchical principle * Non hierarchical principles * Mixed principles	Stimulated by: * New needs * Student feedback * Ideological consideration * Professional consideration	Demonstrated changes in: * Social * Political * Other involvement in content issues

* Formal notations are omitted from this preliminary presentation

Table 8: A Faceted Design for the Characterization of IT Software Ecologically Sensitive Experimentations

FACET	FACET ELEMENT
Type of Information	1. Subject matter bound 2. Procedural
Cognitive Demands	1. Low level (reactive) 2. Low level supplemented by a limited number of higher level demands 3. High level & interactive (reflecting on and constructing own learning processes)
Proximity to Existing Learning Processes	1. "More of the same" (not augmenting "zone for proximal development") 2. "Different & complementary" (augmenting "zone for proximal development")
Exposure to Alternating Representations	1. Use of software in a non selective manner 2. Use of software in a selective manner

Table 9: A Stepwise Design of an Ecologically Sensitive Experimentation

STEP	TREATMENT	CONTROL	POST TEST
PRE CONDITIONS	Equivalent: IQ & score distribution Equivalent teacher evaluations		
STEP 1	Teacher trained in subject matter concepts & in a mindful use of the specific software	Teacher trained in subject matter concepts	Skilled & selective use of software in tentative classroom session (treatment only)
STEP 2	Students trained to mindfully use the specific software with familiar subject matter	Students learn identical & familiar subject matter in a different way	Skilled use of software (treatment only)
STEP 3	Software used by teacher & student with curricular subject matter	Student learn identical subject matter without the software	Achievement test (treatment & control)
STEP 4	Software used with personal assignment	Personal assignment prepared without software	Tests: Abstraction & application of basic principles of the learned subject matter; Learning modes; Perception of learner & teacher role

Table 10a Instructional Decision Making Facets and their Elements

FACET	ELEMENT
[A] Goals of Instruction*	1. Knowledge & comprehension 2. Application 3. Analysis & synthesis 4. At least two of the above
[B] Responsibility for the design of the content to be learned (who decides ?)	1. Curriculum expert 2. Teacher 3. Student 4. Two or all of the above
[C] Type of learning activities that teacher decides to enhance**	1. Teacher directed activities 2. Group directed activities 3. Individualized & autonomous activities 4. At least two of the above activities
[D] Enhanced learning styles***	1. Reaction to uniform structured learning materials 2. Interaction with peers & learning materials 3. Rediscovery & construction of new knowledge 4. Learning in more than one of the above modalities
[E] Teacher's perception of her pedagogical role****	1. Dominating 2. Facilitating

* The elements of Facet [A] are specified in accord with Bloom's Taxonomy (Bloom, 1956)

** Types of learning activities that teachers decide to enhance (Facet [C]) are discussed by Clements (1989)

*** Facet [D] combines learning styles distinctions and concepts discussed by: Henson & Borthwick (1984); characteristics of interactive learning defined by Vygotsky (vygotsky in Wertsch, 1985 and applied by Tharp & Gallimore (1988), and discovery and constructivism notions elaborated by: Bruner, 1966; Give'on, 1987; Clements,1989; Cohen, 1988; Papert, 1990

**** The distinctions presented in Facet [E] are based on Lamm (1976) and on Bennet et al. (1976)

Table 10b: Faceted Specifications of Software Properties and Software Classification

FACET	ELEMENTS	SOFTWARE
MANAGERIAL PROPERTIES [F] Software control devices	1. Control devices pre-programmed into software	Drill & Practice Tutorial
	2. Software partially controls user's activity	Simulation Games
	3. Software is not equipped with a user's device	Open tools Ed. programming
[G] Record keeping*	1. Programmed record keeping provides automatic feedback to user	Drill & Practice Tutorial
	2. Software provides optional record keeping	Games Simulation
	3. Record keeping device is not available	Open tools Ed. programming
INFORMATION PROCESSING** [H] Provision of Information being processed	1. Information is entirely provided by software	Drill & Practice Tutorial
	2. Information is jointly provided by software & user	Games Simulation
	3. Information is provided by user	Open tools Ed. programming
[I] Relation between software & content being processed	1. Structured: curricular content & software cannot be separated	Drill & Practice Tutorial
	2. Semi-Structured content provided by software	Games Simulation
	3. Software is content free	Open tools Ed. programming
[J] Users' involvement in information processing***	1. Software directed information acquisition	Drill & Practice Tutorial
	2. User is required to discover & react to a software's algorithm	Games Simulation
	3. User is required to use software's algorithm & construct content	Open tools Ed. programming

* Feedback to learners and teachers concerning the student's

progress
is frequently discused in the educational
literature(e.g.,Clements,
1989; Ediger, 1988; Give'on, 1988)

** The distinctions provided by Facets [A],[B] and [C] elaborate Give'on's Taxonomy of Educational Software (1988)

*** The assumption that directed activities exclude generative information processing is based on Wittrock's theory of generative thinking (Wittrock, 1974)

Table 11: Mapping Elements of Patterns of Instruction on Elements of Software Properties

PATTERN OF INSTRUCTION PROPERTIES	SOFTWARE PROPERTIES**											
	Software Control Devices			Provision of Info. being Processed			Relation Between Software & Content			Users' Involved in Info. Processing		
	1	2	3	1	2	3	1	2	3	1	2	3
Goals of Instruction												
1 Knowledge & Comprehension	PT	MS		PT	MS	L	PT	MS		PT	MS	
2 Application		MS	LSR		LS	LS		MS	LR		MS	LR
3 Analysis & Synthesis			LSR			LSR			LR			LR
4 At least two of the above*												
Curricular Decisions (Who decides ?)												
1 Curriculum expert	PT	MS		PT	MS		PT	MS				
2 Teacher		MS			MS			MS	L			
3 Student		MS	LSR		MS	LR		MS	L			
4 At least two of the above												
Learning Activity												
1 Teacher directed	PT	MS		PT	MS		PT	MS	LR	PT	MS	LR
2 Group directed		MS	LR		MS	LR		MS			MS	LR
3 Individual & autonomous	PT	MS	LR	PT	MS	LR		MS	LR	PT	MS	LR
4 At least two of the above												
Enhanced Learning Style**												
1 Reaction to structured learning materials	PT			PT	MS		PT	MS		PT		
2 Interaction with peers & learning materials		MS			MS			MS			MS	
3 Rediscovery & construction of knowledge		MS	LR		MS	LR		MS	LR			LR
Perception of Teacher's Pedagogical role												
1 Dominates	PT	MS		PT	MS		PT	MS		PT	MS	
2 Facilitates		MS	LR		MS	LR		MS	LR		MS	LR

Types of Software: P - Drill & Practice; T - Tutorial;
M - Games; S - Simulation;
L - Open Tools; R - Programming

* Cross classifications of software properties and a combination of instructional elements depend on the specific combination selected

** The numbers (1,2,3) refer to the elements of the facets

Appendix 1: Comptown's Operational Principles

1. "High density" allocation of computers enabling each student to have access to a computer for two to three hours a week in a computer laboratory as well as in the regular classroom.
2. Integration of varied computer application into as many subject matters as possible focussing on "open tools" rather on "closed" and "structured" software.
3. A system approach selecting ways and means aimed at the whole educational system (i.e. all the participants and all the activities), as well as its close environment.
4. A multimedia approach: where participants interacting with diverse representations of the same task could change the course of learning and instruction and develop exploratory processes.
5. Use of computer tools in a mindful manner: where students and teachers could be encouraged to reflect on their work and develop new understanding of the learned task.
6. Involvement of the classroom nesting systems (school, community, ministry) in the project's making and implementing policies.
7. Cultivation of positive attitudes and beliefs towards the project within and across the systems that create the ecological environment in which the innovation is designed to take place.

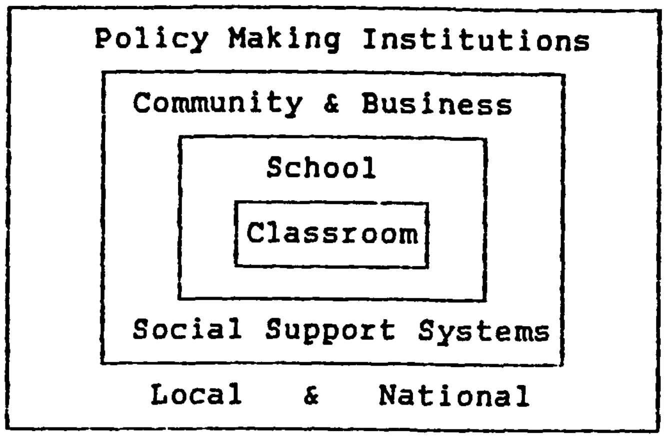


Figure 1: Schematic of the Innovation's Ecology

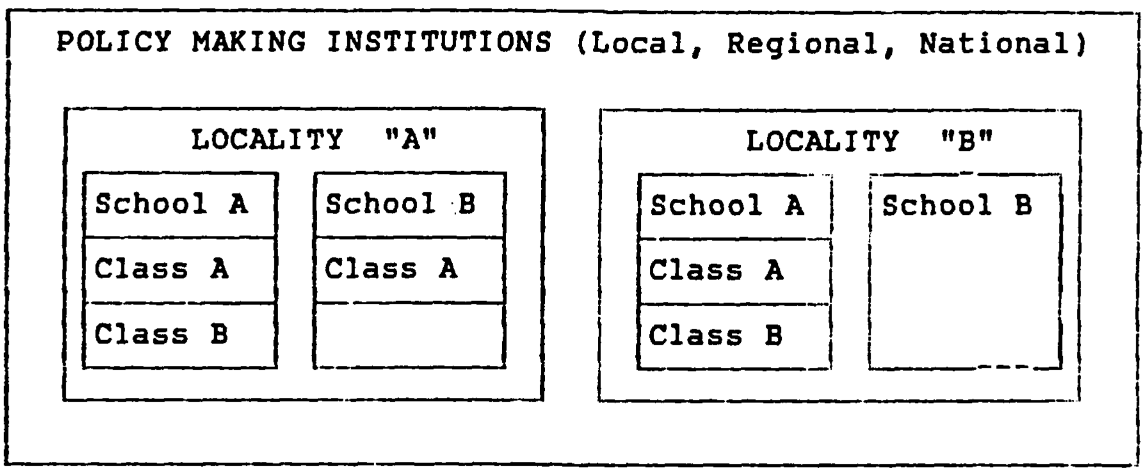


Figure 2: Ecological Units of Analysis Used in Comptown

Assessment Of Distance Learning Technology

Richard E. Clark
University of Southern California

Prepared for delivery at the Conference on Technology Assessment: Estimating the Future. September 5-6, 1990, UCLA Center for Technology Assessment.

Abstract

The four goals of this discussion are to: 1) Discourage distance learning evaluation questions and tactics which have not proved useful in the past. 2) Persuade distance learning evaluation designers to distinguish between the effects of two distinctly different technologies; delivery and instructional technology. 3) Offer brief descriptions of evaluation plans, questions and examples associated with delivery technologies on the one hand, and instructional technology on the other, and 4) discuss issues related to the cost effectiveness evaluation of distance learning.

Introduction

The history of distance learning technology claims (1, 2, 3) and counter claims (4, 5, 6) have led many to accept the need for a change in the way we evaluate new technologies for distance learning. The number of new and complex technological devices that could be applied to distance learning is increasing. Cuban, in his recent book Teachers and Machines (2), cautions that "determining what levels of [technology] use now exist is like trying to snap a photograph of a speeding bicyclist".

With adequate evaluation in place, we may be able to "tune" existing technologies so that they meet our needs, anticipate new developments, and settle disputes, in time to plan and operate rational, cost-effective K-12 local, regional and national distance learning systems. Underlying all evaluation plans are beliefs about how we employ technology so that we enhance the delivery of instruction and the quality of learning experiences. At the heart of every evaluation plan is a curiosity about how new technologies can increase student access to quality instruction and thereby increase their academic achievement, motivation and value for learning.

The purpose of this discussion is to a) discourage evaluation questions which have not proved useful in the past, b) suggest that future evaluations distinguish between the effects of delivery and instructional technologies, c) offer some generic evaluation plans, questions and examples associated with delivery and instruction; and

d) discuss issues related to the evaluation of distance learning cost-effectiveness.

I. Forming and Asking Evaluation Questions

Evaluation is the process by which we judge the "worthwhileness" of something. Since our values govern all evaluation activities, we need to be clear about the kinds of distance learning evaluation questions that will meet the needs of our schools and communities. The questions we decide to ask about distance learning and the evaluation instruments we employ will necessarily keep us ignorant about some matters while informing us about others. Evaluation questions carry implicit assumptions and beliefs about the significance of different elements of distance learning and their impact on desired outcomes. For example, if we ask whether a new teaching medium produces more student achievement than traditional media, we have assumed that media are able to influence student achievement -- an assumption which has been seriously questioned (4,5,6).

One of the most important recommendations underlying this discussion is that all evaluations should explicitly investigate the relative benefit of two different but compatible types of distance learning technologies found in every distance learning program. One technology influences the delivery of schooling and another technology influences instruction. These two technologies are typically confused in most distance learning evaluations. Technology benefits caused by instructional technology are attributed to delivery technologies, and vice versa. The confusion of technological benefits can lead to inappropriate policy decisions. At the root of the confusion one finds different definitions of "technology".

A. Which Technology for What Purpose?

It its most general sense, The term "technology" suggests the application of science and experience to solving problems (8). The major obstacle in our past struggle to understand the contribution of new technology in distance learning is that we have confused the contributions of these two different technologies.

One distinct class of technologies results from the application of various scientific and engineering principles to hardware that records and transmits instruction. These "media" technologies are associated with the physical sciences that have produced the new electronic media (e.g. fiber optics, television, computers). Delivery technologies increase student and teacher access to learning resources which is one of the most important goal of distance learning.

A second type of technology applies various social science principles to suggest teaching methods and curriculum choices. This instructional technology draws primarily on research in teaching, learning and motivation to enhance student achievement. The "products" of an instructional technology are new instructional design theories (9), teaching methods and motivational strategies (4) which can be embedded in "courseware" (instructional materials) for distance learning. One purpose of this discussion is to recommend that all evaluations of distance learning programs attempt to provide reliable and valid determinations of the separate influence of delivery and instructional technologies.

B. Separating Delivery and Instructional Questions

Support for a separate consideration of delivery and instructional technologies in evaluation is well established in the research literature but rare in evaluations or program planning. Wilbur Schramm, the most established reviewer of media studies in education, concluded (10) that "...learning seems to be affected more by what is delivered than by the delivery medium" (p.273). For the past two decades at least, most of the exhaustive analyses of research that compared the learning benefits of different media (4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23) could be summarized with the analogy that media "... do not influence learning any more than the truck that delivers groceries influences the nutrition of a community" (11, p.3). Distance learning media are vehicles that transport instruction to students. The choice of vehicle influences the important outcomes of student access, and the speed or cost of the delivery but not the learning impact of the instruction that is delivered to the "consumer". Delivery vehicles indiscriminately carry helpful, hurtful and neutral instruction.

II. Choosing Critical Indicators For Distance Learning Technologies

Among the specific issues that must be addressed in future evaluations of local applications of distance learning technology are: "What aspects of evaluation planning enhance the usefulness of information for decision makers?", and "How might we collect information which will aid in our judgement about the different influences of the delivery and instructional features of the program?" (12, 21, 24). While a number of evaluation concerns apply to some but not all programs, three generalizations seem useful to all, 1) Adopt an early concern for evaluation, 2) Use a multi-level evaluation plan, and 3) Conduct formal cost-effectiveness analyses.

A. Adopt An Early Concern For Evaluation

Both evaluation specialists and administrative decision makers need to be involved early and actively in distance learning system design. Past experience suggests that waiting until a system is designed before thinking about evaluation has been common but very wasteful. It is critical to have early information about, for example, exactly what set of conditions are being replaced by new distance learning programs. One way to accomplish this would be to spend an ample amount of time during the program planning stages to carefully describe the specific problems we wish the new approach to solve. We should describe how we will measure the current conditions (e.g., a base-line measure of the existing situation --including the views and impressions of the "stakeholders") and thoroughly discuss what we believe to be the alternative solutions to the problem(s).

If an evaluation plan is developed as the program is planned and implemented a number of advantages are realized. In the area of computer assisted learning, Henry Levin (25, 26, 27) describes eight exemplary cost-effectiveness evaluation programs. Each of these good examples collected baseline measures of the problems they were trying to solve. Each of the eight programs began their concern with evaluation at the start of their planning.

Early evaluation makes it possible to determine which aspects of a distance learning program were positive and which were negative. Negative aspects can then be modified and the positive accentuated to achieve maximum benefit. For example, most distance learning programs attempt to bring a much richer set of curriculum choices and quality teaching to K-12 school programs. Program planners might begin with an analysis of options (the variety of media available to deliver new curricula) and their audience (e.g. measure the number of "students" who would enroll in new courses). Early concern with evaluation results in the collection of information on both the need and the audience for distance learning as well as the existing alternatives.

Another advantage of early evaluation involvement is that an ongoing evaluation plan can be developed. Too often programs are developed, implemented and at some later stage the program planners remember that "we have to do some evaluation". As a result, very little is learned about the program being studied which is useful either for the immediate program or for other distance learning ventures. It is early evaluation planning most often yields a useful information and yet it is a rare phenomenon. Levin noted that his search for adequate cost effectiveness evaluations was difficult. He found that only one in six published reports were adequately designed. Since most reports are not published, one suspects that early evaluation is not our typical procedure at the moment.

The second general direction that is useful for all distance learning programs is to adopt a multi-level evaluation plan.

B. Use a Multi-Level Evaluation Plan

The two levels of evaluation that most often seem to give useful development information are to measure: 1) participant reactions, and 2) the achievement of program objectives.

1) Participant reactions to distance learning program effectiveness is the most common (and unfortunately, often the only) level of evaluation attempted by K-12 school districts. Typically, this level of evaluation employs printed forms containing a combination of questions designed to inquire about the "feelings" and "impressions" of different groups who are involved in the program. A common question is "How would you rate the quality of the teaching in this program?" (typically rated on a five point scale that ranges from Exceptional through Average to Poor). Items such as "List what you think are the STRONG [or WEAK] points of the program?" permit the respondent to write in personal views and comments. Questionnaire forms are most often used for reactions because they protect the anonymity of respondents and therefore, we presume, increase the candor of the responses. Forms are often sent to all of the program participants but are filled out and returned by only a small percent of those who receive them. Participant Reactions are useful provided that they do not serve as they only level of evaluation data. They should be used primarily to uncover both informal participant impressions and unanticipated benefits and problems. Reaction

items should be divided between those that deal with the medium (e.g., ease of access, reliability or technical quality of transmission or machines, space allocation issues) and those associated with the instruction (e.g., the quality of teaching, how things learned in the program were used outside of class).

The advantage of collecting participant reactions is that program mangers get informal impressions of the programs and often uncover unanticipated results. For example, the Northeastern Utah Telelearning project, which uses microcomputer audiographic instruction transmitted between remote schools over telephone lines, found an unexpected problem because they used open-ended reaction forms. Students complained that in the early stages of the program it was very difficult to contact a teacher to get help when it was needed. On the other hand, the Interact Instructional Television Network in Houston Texas used a similar instrument and discovered an unexpected positive outcome of their project. It was observed that students in small television reception rooms tended to help each other a great deal during the instructional program. They could help while the program was continuing without disrupting the teacher or other students. This "peer tutoring" seemed to be having a positive impact on student learning and motivation. Upon closer inspection the tutoring activity seemed to be due to the fact that the microphone which the students used to communicate with the teacher at another location had to be turned on to function. When the microphone was turned off, the students could consult among themselves before they turned it on to answer a question or discuss a point with the teacher. When following up on the peer tutoring finding, the Houston project uncovered the fact that some of the tutors hired to supervise the student television reception rooms were demanding that the students "keep quiet" which discouraged the peer tutoring. The tutors had assumed that talking indicated a "discipline problem" and had to be corrected by their supervisors. Once discovered, the peer tutoring can be encouraged and its barriers can be eliminated (e.g., though adjusting the training of tutors in the Houston example).

The disadvantages of participant reaction data is that it is seldom gathered in such a way that it can be considered either a reliable or valid reflection of the program. This problem is not serious. Unreliable information can still provide useful information as it did in the Houston project. However, questionnaire data can be representative if evaluators select a random sample of participants large enough to engage a meaningful number of each group involved in the project. To increase participation evaluators have found it useful to send each randomly chosen participant a card telling them that they have been selected, that their response is vital and to expect the questionnaire soon. Follow up notes to all those chosen can encourage laggards to send in their forms without violating anonymity. Depending on the numbers involved in the entire program, a small (five to ten percent) random sample of participants can give a very accurate impression of the reactions of the entire group.

Questionnaires should be used at various stages in the program development, including very early on. Unanticipated problems and benefits uncovered by questionnaires usually require much more careful study. For example, when students in most distance learning programs are asked, a majority will typically state that they would not continue to elect a distance learning option if they could choose a "traditional class" as they did in the Northeastern Utah Telelearning Project. The fact that students would elect a traditional program if one was offered does not indicate that the Utah project failed. Upon closer inspection it is often found, as is suspected in the Utah case, that students sometimes feel isolated in distance learning settings and would therefore select more traditional options for social and "nonacademic" reasons. This is particularity true of middle and high school students. They typically have strong social needs which are not always met in distance learning programs.

Other problems which can be spotted using the "early warning system" of reaction questionnaires are communication problems between participants, the extent and impact of technical difficulties, inappropriate implementation of plans and opportunities to extend the program into new areas. Yet in even the best of circumstances, reaction forms will not give solid information about the achievement of most delivery and instructional goals. For this purpose, programs need to adopt a second level of evaluation.

2) Achievement of Program Objectives is the second and most substantive evaluation goal. Formal measurement of objectives is usually considered by evaluation specialists to be the most crucial information to be gathered. Objectives should be

divided into at least two categories, those associated with delivery and those associated with instruction. One category of outcome which is common to both types of technology is cost-benefit. The discussion turns next to outcomes specific to instructional technology, then to delivery technology outcomes and finally to cost-effectiveness measurement.

Instructional Technology Objectives include student learning, motivation, transfer of knowledge and values. These important goals are influenced by the "courseware" or instructional programs that are developed and/or chosen and transmitted to distant learners. In most cases, instruction is designed by teachers. In some cases, already developed courseware is purchased and transmitted to remote sites. The instructional decisions that are embedded in each lesson influence student learning and motivation. Different teaching method and curriculum options have very different effects on student learning which might be explored in evaluation. So, distance learning evaluation might include at least the following four types of questions related to instructional technology.

> **"Which of the curriculum and teaching method choices in a given distance learning program impacted student achievement and subsequent ability to use (transfer) the knowledge acquired outside of the instructional setting?"**

Achievement can be tested with teacher or publisher achievement tests. Increasingly schools are interested in the extent to which students transfer what they learn outside of school. Transfer might be estimated by open ended questions on reaction forms. If the school district has other schools receiving similar curricula from different delivery forms, an obvious opportunity exists to check on any achievement or motivation differences between the options. When possible, alternative teaching methods and curriculum choices should be explored in order to maximize the learning of different kinds of students. For example, highly structured and supportive instruction might be contrasted with a more "learner directed" and discovery approach to curriculum (12). Many programs have found that students who are anxious or have learning problems profit a great deal from added structure and support. Whereas students who are more independent and able tend to benefit more from a discovery approach (11).

> **"What impacted student and teacher motivation to learn and invest effort**

in making this program a success?"

Current theories of motivation have introduced a very novel element in distance learning programs and evaluation. Formerly, it was thought that media choices greatly influenced both student and teacher motivation. Now it is understood that motivation is influenced by beliefs and expectations and is therefore due to "individual differences in beliefs about media" and not to the media per se (5, 11, 22). Yet, it is a common belief that students are excited and teachers are threatened by new media (2). There is recent but solid evidence that when students expect that a new medium will make learning easier and more "entertaining", they like it. However, there is good evidence that their liking does not lead them to work harder (3, 5). Quite to the contrary, the more they think a medium makes learning "easy" the less effort they will invest to learn (18, 22). This effect has been explained as a misjudgment about the kind of effort that is required to learn based on our previous experience and expectations. For example, American students typically assume that television is an "easier" medium than books or teachers, probably because of their use of the medium for entertainment. This reaction on the part of our students is quite different than that of Israeli students who, on the average, have been found to invest more effort in television because their early experiences with television have been less entertaining and more demanding intellectually (22).

There is additional evidence that students will not invest effort if they believe a medium to be very difficult. With American children, this is sometimes the reason for their lack of willingness to read (18, 22). So the greatest motivation is invested in media and instructional programs that are perceived as being moderately difficult. This evidence would suggest that one way to influence student motivation would be to select "moderately difficult" media. However the evidence also suggests that student and teacher beliefs about media difficulty change over time, sometimes radically (5). The more stable predictor of motivation seems to be student beliefs about their own ability and the demands placed on them by different instructional tasks (18). This would suggest that we should evaluate the students perceptions and beliefs about the learning tasks contained within the media employed by distance learning programs and their own self efficacy as learners. This form of evaluation could be embedded in reaction questionnaires.

"Which of the curriculum and teaching method choices in a given distance learning program impacted student and teacher values for what was learned and subsequent motivation to teach and learn and to use what was learned outside of the instructional setting?"

Reaction questionnaires which are carefully constructed and administered will give a good indication of student and teacher values related to the program, teaching and the curriculum. Negative value statements do not always reflect negatively on the program (recall the students in the Utah project who liked traditional classrooms better than distance learning because of social opportunities). Generally one hopes to foster a positive value for learning and new curriculum options with distance learning. Shifts in attitude that result from changes in the program can be monitored if reaction forms are sent periodically (every few months) throughout the development stages.

"Which of the curriculum and teaching method choices in a given distance learning program impacted the cultivation of different kinds of knowledge including procedural skills and higher order thinking, learning-to-learn and metacognitive skills?"

While higher order skill learning is more difficult to assess than ordinary "achievement", some programs have been successful in this area. Perhaps the most exciting current example in wide use is to be found in the HOTS program developed by Dr. Stan Pogrow at the University of Arizona (24). HOTS (Higher Order Thinking Skills) is a successful and widely disseminated program for Title I students. Teachers in the program use computer lessons, class exercises and discussion to increase the thinking and study skills of students. Evaluation involves the ongoing use of standardized tests, noting changes in the quality of questions students ask and analyses of their class assignments. While a few formal measures of thinking and study skills exist (and more are being developed), program managers might consult with evaluation specialists about selecting and developing tests to measure problem solving and study skill development (24,25).

While learning, values and study skills are important instructional outcomes for distance learning, the delivery technology will influence yet another type of outcome.

Delivery Technology transmits various forms of instruction to students. The recent introduction of computers to schools has resulted in more attention to technology delivery benefits (15, 24). Evaluation questions associated with delivery technologies include attempts to assess the effect of medium on 1) student access to a greater variety of curriculum choices, 2) school or program utilization of resources, and 3) the reliability of delivery choices. Questions one typically finds in the evaluation of media include:

"Did the distance learning media maximize student access to new, and/or high quality courses and teaching when compared with other choices?"

Access to new or beneficial courses and instructional techniques or teachers is one of the primary objectives of most distant learning programs. Collecting access data often involves comparisons between different ways to deliver courses or the size of enrollments in classes both before and during the implementation of the program. For example, the Share-Ed program in Beaver County Oklahoma used a new fiber optic network to provide new curriculum to rural schools. They collected participant reactions on the advantages of the increased curriculum choices offered to students who allowed to take college credit courses in high school as a result of the new system. These reactions, when combined with baseline and process data on actual enrollments, provide good evidence of the extent of access provided by the innovation. Evaluators should carefully consider increased or enhanced access of minority, older or widely dispersed student groups.

While "access" usually suggests the availability of new curriculum options, it can also imply teacher access to students on a more personal level. Teachers in the Houston Texas InterAct Instructional Television systems report problems with their personal and immediate access to students during instruction in order to "check their reactions or mood" and adjust their teaching accordingly. Whereas teachers using computer delivered courses often report increased "individualized" access to students and enjoy the opportunity to "watch them learn".

"Did the media influence the utilization of school and community educational resources (e.g.space, equipment, skilled teachers, new

courseware developed at one site but not readily available at others)?"

It is often the case that because distance learning programs are recorded and distributed to many different sites, the best teachers are made available to many more students. Evaluators might track statistics about how the background and/or training of teachers in distance programs compare with district averages. An instance of a different kind of utilization is to be found in the Beaver County Oklahoma Share-Ed program. The local telephone company was installing fiber optic communication lines to improve local service. The system was capable of handling far greater transmission volume than the existing usage anticipated in the communities served. The school system's use of fiber optic lines for television and voice transmission for distance learning utilized unused space on the system. Since distance learning courses are often provided to fewer students per school than the average course, they often make use of under utilized rooms (e.g. storage spaces) and equipment.

"Are distance learning media more reliable than other alternatives?"

One of the primary concerns expressed by the critics of distance media is their technical reliability. In the Beaver County Oklahoma television system for example, the reaction forms used in evaluation only picked up technical problems when the students were asked to describe "weak points" of the system. None of the administrators noticed technical problems, eleven percent of the teachers mentioned reliability, but thirty six percent of the students responded to the reaction form by going into detail about microphone feedback, distracting equipment, out-of-focus pictures, equipment noises and color problems. This difference in reporting reliability problems probably stems from the amount of experience each group had with the actual television transmission. However, program evaluation should establish regular checks by technical staff on these problems in order to judge the severity of participant reactions and make repairs when necessary. When technical transmission problems are not solved, they can decrease achievement scores and reduce participant commitment to the system.

In all successful distance learning programs, delivery (media) technology and instructional technology must work together. The delivery features of new media must be employed so that they will eventually save precious educational resources. Curriculum and instructional design must be utilized so that they support the effective learning and transfer of important concepts. Instruction must be developed to reflect the special delivery characteristics of different media. In addition however, communities and funding agencies are increasingly concerned not only with the effectiveness but also the cost of distance learning programs. Cost is a "goal" or "outcome" of both delivery and instructional technologies.

III. Cost-Effectiveness Evaluation

During an evaluation of the separate delivery and instructional value of distance learning program effectiveness, cost data should also be collected. This parallel activity allows us to combine "effectiveness" (i.e., delivery and instructional outcomes) with "cost" data to provide cost-effectiveness information to decision makers.

In many ways, cost-effectiveness ratios are the most interesting information we can supply to school officials, taxpayers and their elected representatives. Limited educational resources will eventually require a much greater emphasis on both the monetary and time cost of new programs.

A. Delivery Technology Cost

Evaluations that precede the introduction of new media should explore the costs of various alternatives. In many cases, older technologies (e.g., tutors, books, cassette television programs, the mail system) are cheaper in monetary cost but very "expensive" in delivery time and reliability. Evaluations of costs should always consider trade-offs with cheaper and more traditional delivery options. There is evaluation data which indicates, for example, that tutors who are trained and paid minimum wage are much cheaper than computers for some instructional purposes (25).

Evaluations which are conducted during the introduction and maintenance of a distance learning program are advised to adopt the "ingredients" costing approach described below.

B. Instructional Technology Cost

There are a great variety of different school and community goals that influence evaluation criteria under the general heading of instructional effectiveness costs. The cost involved in increasing student motivation, learning and transfer are being questioned with greater frequency. School districts may wish to consider collecting cost data which will aid policy makers. The development of an instructional technology yields a variety of teaching, motivation and transfer outcomes at very different monetary costs.

Besides monetary cost, schools are increasingly interested in the time costs associated with the mastery of different learning or performance goals. Some types of learning tasks consume much more "teaching time" and/or "learning time"(5). For example, it takes much longer to teach a student study skills than to teach memorization of facts. It also takes longer for a student to learn procedural knowledge to the point where it becomes automatic -- about 100 hours of practice for even simple procedures is the current estimate. Therefore, there will be more and more emphasis on the time costs of different instructional technology options. In many areas, the cheapest option is not necessarily the best. In the same way, the quickest option among instructional technologies is not always the best. Students who learn faster do not necessarily learn better. The new "cognitive" learning theories provide the insight that it may be more important to know <u>how</u> students reach learning goals than to know that they get correct answers on examinations. It often takes longer for students to learn in such a way that their correct answer on a test reflects "deep cognitive processing" and the exercise of "higher order cognitive learning skills", than to take a surface level shortcut. Educators need to be wary of focusing evaluations on time savings at the expense of the quality of learning.

Generally, once a distance learning team has worked out the list of goals associated with both monetary and time costs, an evaluation design can be chosen. One of the first issues to be confronted is the choice of how the data reflecting costs will be gathered. While there are a number of methods, one seems particularly applicable to both delivery and instructional technologies - Levin's (25, 26, 27) <u>ingredients</u> method.

<u>The Ingredients Method of Determining Costs</u> While there are a number of emerging ways to determine local costs and efficiencies, one of the soundest and most comprehensive is the "ingredients method" developed by Henry Levin at Stanford University (25, 27). It " requires identification of all of the ingredients required for the ...

[distance learning] intervention, a valuation or costing of those ingredients and a summation of the costs to determine the cost of the intervention" (27, p. 3). In the K-12 setting, cost is defined as the value of what is given up by using resources in one way rather than for its next best alternative use. For example, if teacher time is given up then it may not be used for other purposes. Therefore, the cost of teacher time is assessed by assigning a value to what is lost when teachers are assigned to distance learning technology programs.

The ingredients method is implemented in two stages. In the first stage, all necessary program ingredients are listed. The identification of ingredients requires that we list distance learning program necessities associated with five categories: 1. personnel, 2. facilities, 3 equipment, 4. materials and supplies and 5. all other. In the second stage, each of the ingredients listed in each of the five categories is valued.

Space limitations preclude a complete description of the ingredients method but a review of Levin (26,27) will provide most of the information needed to determine ingredient costs. Levin gives specific technology examples which are very relevant to the kinds of programs now evolving in many schools and he urges complete listings of ingredients. For example, he requires that all "donated" time of volunteers and outside organizations be included as a personnel ingredient if it is necessary for the conduct of the program. He reasons that failure to cost donated time will give an unrealistic picture of the "replication" expense. He also claims (26) that, in the rare instance where one finds a complete costing of technology-based programs, one often finds evidence that the organizational climate greatly influences cost-benefit ratios. Some organizational plans seem to be much more efficient than others.

IV. Conclusion

In the past, distance learning evaluations have typically been conducted as "afterthoughts" and have relied heavily on reaction questionnaires which are unreliable and nonrepresentative of the participants involved. Even when evaluations attempted to collect information about changes in student achievement, questions were asked which confused the separate contributions of delivery media and instructional technology.

In order to identify the strong features of distance learning programs and eliminate weak features, more robust evaluation plans must be adopted in the future. These plans should be firmly based on the experience of those who have struggled with technology evaluations in the past (27). Three features are recommended: First, evaluation should begin at the start of distance learning program planning. An early commitment to evaluation will provide much more useful information about the strengths of a program as it develops. Changes can be made during the formative stage in time to strengthen the plan. The second recommendation is that all programs should adopt a multi-level evaluation plan. The different role of qualitative (e.g. questionnaire) and quantitative (e.g. student achievement scores, monetary costs) data should be decided. Delivery and Instructional evaluation should be separated and a variety of goals assessed. Finally, new techniques are available for cost-effectiveness evaluation of distance learning programs. Levin's "ingredients" method is suggested.

References

1. Benjamin, L.T., (1988) A history of teaching machines, American Psychologist, 43(9), 703-712.

2. Cuban, L. (1986) Teachers and machines: The classroom use of technology since 1920. New York: Teachers College Press.

3. Saettler, P. A., (1968), A history of instructional technology. New York: McGraw Hill.

4. Clark, R. E. & Salomon, G. (1986) Media in teaching. In Wittrock, M. C. (Ed.), Handbook of Research on Teaching. Third edition. New York:Macmillan.

5. Clark, R. E. & Sugrue, B. M. (1989) Research on instructional media: 1978-1988. In D. Ely (Ed.) Educational Media and Technology Yearbook volume 14. Denver, CO:Libraries Unlimited.

6. Gardner, H. & Salomon, G (January, 1986) The computer as educator: Lessons from television research. Educational Researcher, 13-19.

7. Clark, R. E. (1989), Current progress and future directions in research on instructional technology. Educational Technology Research and Development. 37(1), 57-66.

8. Heinich, R. (1984) The proper study of instructional technology. Educational Communication and Technology Journal, 32(2), 67-87.

9. Reigeluth, C. (1983) Instructional design: Theories and models. Hillsdale N.J.: Lawrence Earlbaum Associates.

10. Schramm, W. (1977). Big media, little media. Beverly Hills, CA:Sage Publications.

11. Clark, R. E. (1983) Reconsidering research on learning from media. Review of Educational Research, 53(4), 445-459.

12. Clark, R. E. (1985) Evidence for confounding in computer-based instruction studies:Analyzing the meta-analyses. Educational Communication and Technology Journal, 33(4), 249-262.

13. Allen, W. H. (1971) Instructional media research: Past, present and future. A V Communication Review, 19, 5-18.

14. Jamison, D., Suppes, P., and Wells, S. (1974) The effectiveness of alternative instructional media: A survey. Review of Educational Research, 44, 1-68.

15. Kearsley, G., Hunter, B., and Sidel, R. J., (1983) Two decades of computer based instruction:What have we learned? T.H.E. Journal, 10, 88-96.

16. Levie, W. H. and Dickie, K. (1973) The analysis and application of media. In R. M. W. Travers (Ed.) Second handbook of research on teaching. Chicago: Rand McNally.

17. Lumsdaine, A. (1963). Instruments and media of instruction. In N. L. Gage (Ed.) Handbook of research on teaching. Chicago:Rand McNally.

18. Mielke, K. (1968). Questioning the questions of ETV research. Educational Broadcasting Review. 2, 6-15.

19. Salomon, G. (1981) Communication and Education. Beverly Hills CA: Sage Publications.

20. Winn, W. (1982). Visualization in learning and instruction. Educational Communication and Technology Journal, 30(1), 3-25.

21. Baker, E. L. (1989) Technology assessment: Policy and methodological issues. Air Force Systems Command, "Proceedings of the 2nd intelligent tutoring systems research forum", 151-158.

22. Salomon, G. (1984) Television is "easy" and print is "tough": The differential investment of mental effort in learning as a function of perceptions and attributions. Journal of Educational Psychology. 76(4), 647-658..

23. Clark, R. E. & Voogel, A. (1985) Transfer of training for instructional design. Educational Communications and Technology Journal, 33(2), 113-123.

24. Congress of the United States, Office of Technology Assessment, Power On: New tools for teaching and learning. (Washington, D.C., U.S. Government Printing Office, September 1988).

25. Levin, H.H. (1983) Cost Effectiveness: A primer. Beverly Hills, CA: SAGE Publications.

26 Levin, H. M. and Meister, G. R. (1985) Educational technology and computers:Promises, promises, always promises. Report No. 85-A13. Stanford, CA:Center for Educational Research at Stanford. School of Education, Stanford University.

27. Levin, H.H. (May 1988) The economics of computer-assisted instruction. In press for a special issue of the Peabody Journal of Education to be edited by J.J. Bosco, Tate Center, Western Michigan University.

28. U.S. Congress, Office of Technology Assessment, Linking for Learning: A New Course for Education, OTA-SET-430 (Washington, DC:U.S. Government Printing Office, November, 1989.

ASSESSMENT OF COMPUTER-BASED INSTRUCTION

James A. Kulik

The University of Michigan

What do evaluation studies say about computer-based instruction? It is not easy to give a simple answer to the question. The term *computer-based instruction* has been applied to too many different programs, and the term *evaluation* has been used in too many different ways. Nonetheless, the question of what the research says cannot be ignored. Researchers want to know the answer, school administrators need to know, and the public deserves to know. How well has computer-based instruction worked?

Reviewers handle such questions in two different ways. Some reviewers are selective in their approach to evidence. They hold that evaluation questions are best answered by key experiments, and so they sift through piles of reports to find the studies with the most convincing results. These studies become the focus of their reviews. Other reviewers feel that evaluation results are inherently variable and that evaluation questions are seldom decided by the results of an experiment or two. Such reviewers put together a composite picture of all the findings on a topic, and they use statistical methods to identify representative results. Both approaches are valuable. The first shows what researchers and developers can accomplish in extraordinary circumstances; the second shows what is likely to be accomplished under typical conditions. We need both types of reviews in the area of computer-based instruction.

All of my reviews on the topic of computer-based instruction, however, have been of the second type. For more than ten years, my colleagues and I have been organizing and summarizing the evaluation literature on computer-based instruction and trying to identify representative results. I believe that comprehensive reviews like ours provide a good context for discussing the more exceptional results in the area. Our reviews provide a background. They make discussions of exceptional results more meaningful because they put them into perspective.

In this chapter, I focus on three aspects of evaluation findings on computer-based instruction. First, I describe the methods that my colleagues and I have used to create a composite picture of findings on computer-based instruction. Second, I present a broad overview of reviewer conclusions, based on nine separate syntheses of the evaluation findings. Third, I take a closer look at a set of nearly 100 evaluations of computer-based instruction in an attempt to reach some more precise conclusions about its effectiveness.

Method

The review method that we use is called *meta-analysis*, and it was given its name by Gene Glass in 1976 in a classic synthesis of the literature on the effects of psychotherapy. Glass used the term meta-analysis to refer to the statistical analysis of a large collection of results from individual studies for the purpose of integrating the findings (Glass, McGaw, & Smith, 1981). Reviewers who carry out meta-analyses first locate studies of an issue by clearly specified procedures. They then characterize the outcomes and features of the studies in quantitative or quasi-quantitative terms. Finally, meta-analysts use multivariate techniques to relate characteristics of the studies to outcomes.

One of Glass's major innovations was his use of measures of *effect size* in research reviews. Researchers had used effect sizes in designing studies long before meta-analysis was developed, but they failed to see the contribution that effect sizes could make to research reviews. Glass saw that results from a variety of different studies could be expressed on a common scale of effect size and that with this transformation reviewers could carry out statistical analyses that were as sophisticated as those carried out by experimenters.

Size of effect can be measured in several ways, but the measure of effect size most often used is the standardized mean difference. Sometimes called Glass's effect size, this index gives the number of standard-deviation units that separates outcome scores of experimental and control groups. It is calculated by subtracting the average score of the control group from the average score of the experimental group and then dividing the remainder by the standard deviation of the measure. For example, if a group that receives computer-based coaching on the SAT obtains an average score of 550 on the test, whereas a group that receives conventional teaching averages 500, the effect size for the coaching treatment is 0.5 since the standard deviation on the SAT is 100.

Methodologists have written at least five books on meta-analytic methods in recent years (Glass et al., 1981; Hedges & Olkin, 1985; Hunter, Schmidt, & Jackson, 1982; Rosenthal, 1984; Wolf, 1986), and reviewers have conducted numerous meta-analyses of research findings. In a recent monograph, for example, we described results from more than 100 meta-analytic reports in education alone (J. Kulik & Kulik, 1989). In addition, meta-analytic methodology has also been used extensively in psychology and the health sciences. Reviewers have used it to draw general conclusions on such diverse subjects as the effects of gender on learning and the effectiveness of coronary bypass surgery.

Overview

At least ten separate meta-analyses have been carried out to answer questions about the effectiveness of computer-based instruction (J. Kulik & Kulik, 1989; Niemiec & Walberg, 1987). The analyses were conducted independently by research teams at four universities: University of Colorado, University of Illinois at Chicago, University of Michigan, and University of Iowa. The research teams focused on different uses of the computer with different populations, and they also differed in the methods that they used to find studies and analyze study results.

Nonetheless, each of the analyses yielded the conclusion that programs of computer-based instruction have a positive record in the evaluation literature.

The following are the major points emerging from these meta-analyses:

1. Students usually learn more in classes in which they receive computer-based instruction (Table 1). The analyses produced slightly different estimates of the magnitude of the computer effect, but all the estimates were positive. At the low end of the estimates was an average effect size of 0.22 in 22 studies conducted in elementary and high school science courses (Willett, Yamashita, & Anderson, 1983). At the other end of the scale, Schmidt, Weinstein, Niemiec, & Walberg (1989) found an average effect size of 0.57 in 18 studies conducted in special education classes. The weighted average effect size in the 9 meta-analyses was 0.34. This means that the average effect of computer-based instruction was to raise examination scores by 0.34 standard deviations, or from the 50th to the 63rd percentile.
2. Students learn their lessons in less time with computer-based instruction. The average reduction in instructional time was 34% in 17 studies of college instruction, 24% in 15 studies in adult education (C. Kulik & Kulik, in press).

Table 1

Findings from 9 Meta-analyses on Computer-Based Instruction

Meta-analysis	Instructional level	Type of application	Number of studies analyzed	Average effect size
Bangert-Drowns, Kulik, & Kulik (1985). Updated in C. Kulik & Kulik (in press).	Secondary	CAI, CMI, CEI	51	0.25
Burns (1981)	Elementary & secondary math	Drill & tutorial	44	0.36
Hartley (1977)	Elementary & secondary math	Drill & tutorial	33	0.41
C. Kulik & Kulik (1986). Updated in C. Kulik & Kulik (in press).	College	CAI, CMI, CEI	119	0.29
C. Kulik, Kulik, & Shwalb (1986). Updated in C. Kulik & Kulik (in press).	Adult education	CAI, CMI, CEI	30	0.38
J. Kulik, Kulik, & Bangert-Drowns (1985). Updated in C. Kulik & Kulik (in press).	Elementary	CAI, CMI, CEI	48	0.37
Niemiec & Walberg (1985)	Elementary	Drill, tutorial CMI, problem-solving	48	0.37
Schmidt, Weinstein, Niemiec, & Walberg (1985)	Special education	Drill, tutorial, & CMI	18	0.57
Willett, Yamashita, & Anderson	Precollege science	CAI, CMI, CSI	11	0.22

Note. CAI = computer-assisted instruction; CEI = computer-enriched instruction; CMI = computer-managed instruction; CSI = computer-simulation in instruction.

3. Students also like their classes more when they receive computer help in them. The average effect of computer-based instruction in 22 studies was to raise attitude-toward-instruction scores by 0.28 standard deviations (C. Kulik & Kulik, in press).
4. Students develop more positive attitudes toward computers when they receive help from them in school. The average effect size in 19 studies on attitude toward computers was 0.34 (C. Kulik & Kulik, in press).
5. Computers do not, however, have positive effects in every area in which they were studied. The average effect of computer-based instruction in 34 studies of attitude toward subject matter was near zero (C. Kulik & Kulik, in press).

This brief review shows that there is a good deal of agreement among meta-analysts on the basic facts about computer-based instruction. All the meta-analyses that I have been able to locate show that adding computer-based instruction to a school program, on the average, improves the results of the program. But the meta-analyses differ somewhat on the size of the gains to be expected. We need to look more closely at the studies to determine which factors might cause variation in meta-analytic results.

Specific Findings

The computer was used in conceptually and procedurally different ways in studies examined in these meta-analyses. Did all the approaches produce the same result? It seems unlikely that they did. It is more reasonable to expect different results from different approaches. A plausible hypothesis is that some computer approaches produce results that are better than average, whereas other approaches produce below-average results.

To examine this hypothesis, I used a set of 97 studies that were carried out in elementary school and high schools (Table 2). Each of the studies was a controlled quantitative study, in which outcomes in a class taught with computer-based instruction were compared to outcomes in a class taught without computer-based instruction. Most of the 97 studies were included in earlier meta-analytic reports on the effectiveness of computer-based instruction (Bangert-Drowns, Kulik, & Kulik, 1985; J. Kulik & Kulik, in press; J. Kulik, Kulik, & Bangert-Drowns, 1985).

There are a number of ways of dividing these studies into groups by computer use. Early taxonomies often distinguished between four uses of the computer in teaching: drill-and-practice, tutorial, dialogue, and management (e.g., Atkinson, 1969). Recent taxonomies collapse some of these categories and add others. Taylor (1980), for example, has distinguished between three uses of the computer in schools: tutor, tool, and tutee. First, as a tutor, the computer presents material, evaluates student responses, determines what to present next, and keeps records of student progress. Most computer uses described in earlier taxonomies fall under this heading in Taylor's

Major Features and Achievement Effect Sizes in 97 Studies of Computer-Based Instruction

Study	Type of Publication	Place	Grade	Course Content	Weeks of Instruction	Effect Size	
						Local	Commercial
Tutoring with Stanford-CCC materials							
Atkinson, 1969	Journal	California	1	Reading	25		----
Cranford, 1976	Dissertation	Mississippi	5,6	Math	12		0.64
Crawford, 1970	Journal	California	7	Math	8		0.10
Davies, 1972	Dissertation	California	3-6	Math	16		0.34
Delon, 1970	Journal	Mississippi	1	Math	36		1.08
Fletcher & Atkinson, 1972	Journal	California	1	Reading	22	0.85	0.79
Jamison, Fletcher, Suppes, & Atkinson, 1976	Journal	---	---	Computer programming	8		0.40
Levy, 1985	Dissertation	New York	5	Reading & math	36		0.22
Litman, 1977	Dissertation	Illinois	4-6	Reading	36		0.23
Mendelsohn, 1972	Journal	New York	2-6	Math	20		0.49
Metric Associates, 1981							
Study I	Report	Massachusetts	1-6	Reading, math & language arts	36		0.32
Study II	Report	Massachusetts	7,8,9	Reading	36		0.56
Miller, 1984	Dissertation	Oregon	5-8	Math	36		0.38
Mravetz, 1980	Dissertation	Ohio	7,8	Reading	12		0.15
Palmer, 1973	Report	California	4-6	Math	16		0.36
Porinchak, 1984	Dissertation	New York	9 - 12	Reading	28		0.19
Prince, 1969	Journal	Mississippi	1-6	Math	36		0.64
Ragosta, 1983	Journal	California	1-6	Reading, Math & Language Arts	108		0.30
Smith, 1980	Dissertation	Washington, D.C.	10	Reading & Math	34		0.33

Table 2 (continued)

Study	Type of Publication	Place	Grade	Course Content	Weeks of Instruction	Effect Size	
						Local	Commercial
Tutoring with Stanford-CCC materials							
Suppes & Morningstar, 1969	Journal	California	1-6	Math	22		0.28
Suppes & Morningstar, 1970	Journal	New York	1-6	Math	18		----
Thompson, 1972	Dissertation	Texas	4-6	Language Arts	36		0.11
Vincent, 1977	Dissertation	Ohio	9-12	Math	10		0.34
Tutoring with other materials							
Alberta Department of Education, 1983	Report	Canada	3	Reading	20	0.60	0.92
Al-Hareky, 1983	Dissertation	Saudi Arabia	4	Math	4		0.56
Anderson, 1984	Dissertation	Nebraska	3 & 4	Keyboard skills	4	0.38	
Bostrom, Cole, Hartley, Lovell, & Tait, 1982	Report	England	8	Math	3	0.16	
Chiang, Stauffer & Cannara, 1978	Report	California		Reading, math & language arts	35		0.19
Cole, 1971	Dissertation	Michigan	10,11,12	Math	16	0.53	0.11
Confer, 1971	Dissertation	Pennsylvania	---	Math	6		0.07
Cooperman, 1985	Dissertation	Delaware	2 - 4	Reading	36		0.04
Diamond, 1969	Report	Pennsylvania	8,9,10	Reading & Biology	36		0.42
Diamond, 1986	Dissertation	Utah	K	Keyboard skills	8	-0.07	
Dunn, 1974							
Study I	Report	Maryland	4	Math	11		0.45
Study II	Report	Maryland	6	Math	30	0.34	0.53
Study III	Report	Maryland	10	Math	36		0.64
Durward, 1973	Report	Canada	6,7	Math	6	0.19	
Easterling, 1982	Dissertation	Texas	5	Reading & Math	16		0.02

Study	Type of Publication	Place	Grade	Course Content	Weeks of Instruction	Effect Size	
						Local	Commercial
			Managing				
Beck & Chamberlain, 1983	Report	Ohio	9	Reading	28		0.16
Broderick et al., 1973	Report	England	---	Biology	7	0.28	
Chamberlain, Beck, & Johnson, 1983	Report	Ohio	4 - 8	Reading	25		-0.15
Coffman & Olsen, 1980	Report	Louisiana	3,4	Reading & Math	72		-0.01
Fisher, 1973	Report	Maryland	10	Geometry	36		-0.26
Larrea-Peterson, 1985	Dissertation	Utah	2 - 6	Reading & math	24		0.60
Nabors, 1974	Dissertation	Missouri	5-6	Problem Solving	36		0.36
Roberts, 1982	Dissertation	Utah	3-6	Reading	108		-0.18
Staniskis, 1977	Dissertation	Pennsylvania	9-12	Biology	36		0.39
			Simulation				
Hughes, 1974	Dissertation	Ohio	12	Physics	8	0.11	-0.46
Jones, 1974	Dissertation	Iowa	11,12	Physics & chemistry	18	-0.10	
Lang, 1976	Dissertation	Nebraska	11,12	Physics	4	-0.09	
Lunetta, 1972	Dissertation	Massachusetts & Connecticut	11,12	Physics	2	0.69	
Melnik, 1986	Dissertation	Illinois	5	Problem solving	10		0.30
Sperry, 1977	Dissertation	Utah	10,11,12	Biology	14		-0.04
			Enrichment				
Ash, 1986	Dissertation	Alabama	5	Math	36		0.48
Boyd, 1973	Dissertation	Illinois	9	Math	14	-0.44	-0.22
Clements, 1986	Journal	Ohio	1 & 3	Reading & math	22		0.29
Coomes, 1986	Dissertation	Texas	4	Reading	36		-0.14
Ferrell, 1985	Report	Texas	6	Math	36		0.39

Study	Type of Publication	Place	Grade	Course Content	Weeks of Instruction	Effect Size	
						Local	Commercial
			Tutoring with other materials				
Fejfar, 1969	Journal	Indiana	4	Math	4	0.80	
Feldhausen, 1986	Dissertation	Nebraska	9 - 12	History	2	0.10	
Gershman & Sakamoto, 1981	Journal	Canada	7-10	Math	18	0.29	
Grocke, 1982	Report	Australia	1-6	Reading	4		0.82
Haberman, 1977	Dissertation	Pennsylvania	4-6	Math	8		0.57
Henderson, 1983	Report	California	9 - 12	Math	12	0.88	
MacLean, 1974	Report	Pennsylvania	3-5	Math	3	0.62	
McEwen & Robinson, 1976	Report	Canada	10	French	10		----
Mitzel, 1971	Report	Pennsylvania	9	Math	36	0.56	0.08
Morgan, Sangston & Pokras, 1977	Report	Maryland	3-6	Math	60	0.23	
O'Connell, 1973	Dissertation	New York	9	Math	2	0.37	
Pachter, 1979	Dissertation	New York	10,11	Math	2	1.44	
Summerlin & Gardner, 1973	Journal	Florida	11	Chemistry	3	-0.38	
Todd, 1986	Dissertation	Texas	4	Reading & Math	28		0.25
Turner, 1986	Dissertation	Arizona	3 & 4	Math	13		0.26
Wainwright, 1985	Dissertation	Minnesota	9 - 12	Chemistry	3	-0.42	
Warner, 1979	Report	Ohio	6	Math	36		1.31
Way, 1984	Report	Kansas	8 - 9	Math	24		0.08
Wilson & Fitzgibbon, 1970	Journal	Michigan	4,5	Language arts	16		0.40
Wilson, 1982	Report	---	10,11,12	Math	18		0.60
			Managing				
Adams et al., 1983	Report	Oregon	2-8	Reading & Math	32		0.18

Study	Type of Publication	Place	Grade	Course Content	Weeks of Instruction	Effect Size	
						Local	Commercial
Programming							
Foster, 1973	Dissertation	Minneapolis	8	Algebra	12	0.39	
Hatfield, 1970	Dissertation	Minnesota	7	Math	36	0.16	0.18
Jhin, 1971	Dissertation	Alabama	11	Algebra	12		0.19
Johnson, 1971	Dissertation	Minneapolis	7	Math	8		-0.60
Katz, 1971	Dissertation	Pennsylvania	11	Algebra	36	-0.01	-0.39
Kieren, 1969	Dissertation	Minnesota	11	Math	36	0.06	0.28
Krull, 1980	Dissertation	Missouri	7	Math	10	0.68	
Mandelbaum, 1974	Dissertation	Pennsylvania	10	Math	20		0.04
Ronan, 1971	Dissertation	Michigan	11	Math	19	0.40	0.20
Logo							
Clements, 1986	Journal	Ohio	1 & 3	Problem solving	22	1.00	
Degelman, Free, Scarlato, Blackburn, & Golden, 1986	Journal	----	K	Problem solving	5	1.20	
Horner & Maddux, 1985	Journal	Texas	Jr. high	Problem solving	6		0.05
Horton & Ryba, 1986	Journal	New Zealand	7	Problem solving	7	0.15	0.46
Odom, 1984	Dissertation	Arkansas	5 & 6	Problem solving	8		0.29
Reimer, 1985	Journal	Canada	K	Problem solving	4	0.75	
Rieber, 1987	Journal	Pennsylvania	2	Problem solving	12	1.52	
Rodefer, 1986	Dissertation	Pennsylvania	4 & 5	Problem solving	8		0.06
Shaw, 1986	Journal	Alaska	5	Problem solving	7		0.13

scheme. Second, the computer serves as a tool when students use it for statistical analysis, calculation, or word processing. Third, the computer serves as a tutee when students give it directions in a programming language that it understands, such as Basic or Logo.

Slavin (1989) has recently advocated a different way of looking at instructional innovations. He believes that innovations can be defined with different degrees of precision. At Level I, innovations are defined vaguely. According to Slavin, such grab-bag categories as open-education and whole-language instruction suggest only fuzzy models for instructional practice. The terms are used for a variety of procedures that do not have a distinct conceptual basis. Level II innovations are more clearly specified. They usually have a conceptual basis that is easy to describe, but in practice Level II approaches are implemented in different ways. Slavin's examples are cooperative learning, direct instruction, mastery learning, and individualized instruction. Level III approaches are precisely defined. They include specific instructional materials, well-developed training procedures for teachers, and detailed prescriptive manuals. Slavin's examples are DISTAR and Man a Course of Study.

Computer-based instruction should probably be thought of as a Level I, or loose, category. The term refers to a variety of procedures with a variety of conceptual bases. It is a chapter heading rather than a technical term. Under this heading, however, fall several well-defined categories of computer use, which can be thought of as Level II categories. An important one is computer-based tutoring. Most programs of computer tutoring derive their basic form from Skinner's work in programmed instruction. Skinner's model emphasized (a) division of instructional material into a sequence of small steps, or instructional frames; (b) learner responses at each step; and (c) immediate feedback after each response. Level III innovations include common instructional materials, training procedures, and so on. One example is the computer-based material developed under the direction of Suppes and Atkinson at Stanford and later disseminated through the Computer Curriculum Corporation.

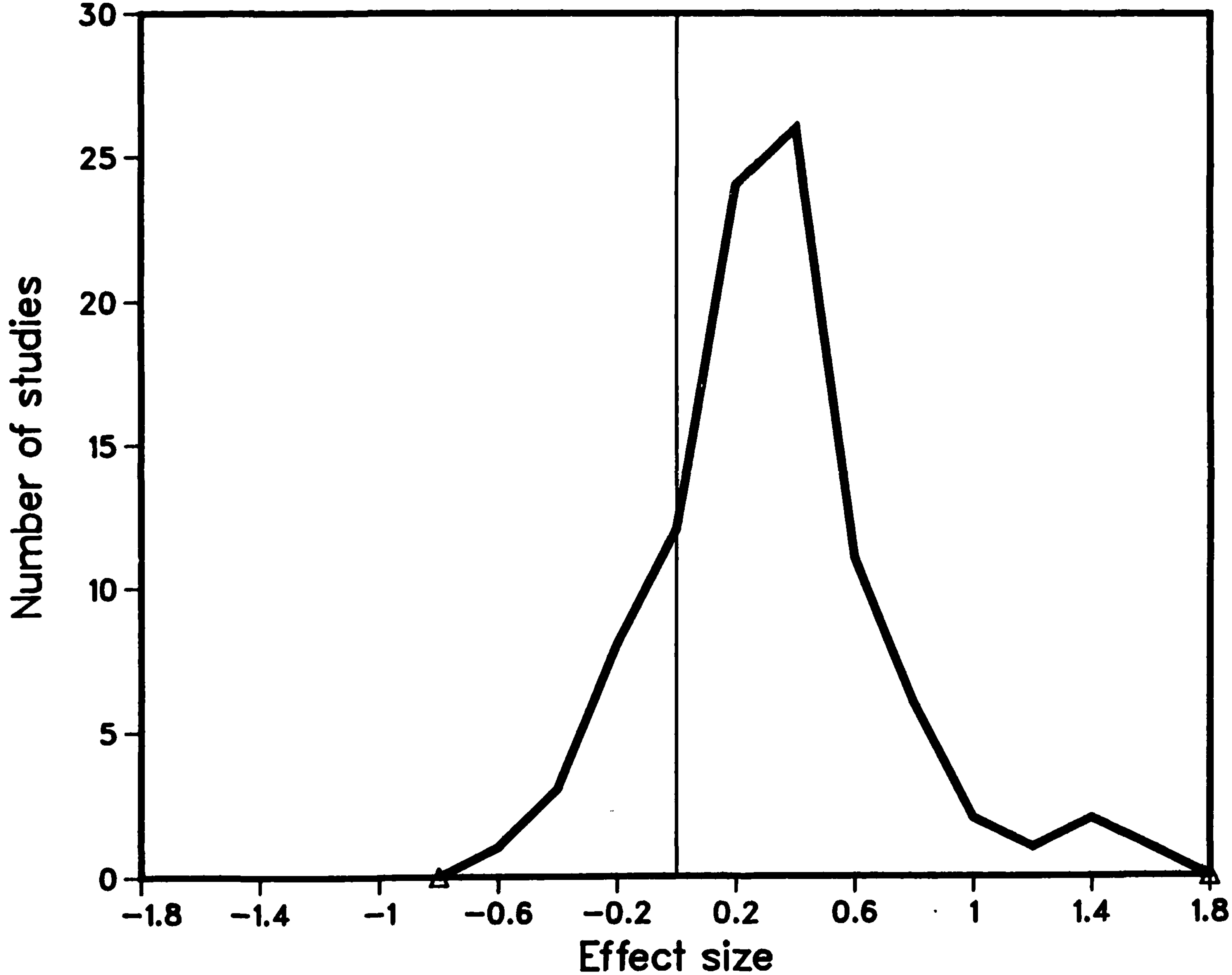
It seems reasonable to suppose that results will be least consistent for the loose category of Level I innovations and that results will be most consistent for Level II and Level III innovations. To examine this hypothesis, I carried out three separate analyses of the 97 studies of computer-based instruction in elementary and high schools. I first examined the effects in all 97 studies. This Level I analysis was broad; it made no concession to the different uses of the computer in different studies. Next, I examined subgroups of studies, grouping the studies by major types of computer use. This analysis was of Level II categories of innovation. Finally, I examined effects in an especially homogeneous subgroup of studies. Each of the studies in this subgroup used similar materials in a similar way. This final analysis focused on a Level III category of innovation.

Level I Analysis

The distribution of the effect sizes is nearly normal in shape (Figure 1). The median of the effect sizes is slightly lower than the mean, however, indicating a slight degree of positive skew in the distribution. This skew is produced by several studies with unusually high effect sizes. Some analysts feel that unusually high and low values are "outliers" that should be eliminated from a distribution. Others believe that extraordinary results merit careful scrutiny because they may provide valuable clues about improving instructional treatments.

The average effect size in the total group of 97 studies, however, is 0.32. This implies that the average student receiving computer-based instruction performed at the 63rd percentile, whereas the average student in a conventional class was at the 50th percentile. Effect sizes can also be interpreted in terms of months on a grade-equivalent scale. Pupils in elementary schools gain approximately 0.1 standard deviations per month in their scores on most standardized tests. An effect size of 0.32 can thus be thought of as equivalent to a gain of about 3 months on a grade-equivalent scale.

Effects of Computer–Based Instruction On Examinations in 97 Studies



The standard deviation of the distribution of effect sizes is 0.39. This implies that approximately two-thirds of all studies found effects between -0.1 and 0.7 and that 95% of all results fell between -0.4 and 1.1. Thus, there is a good deal of uncertainty about the effects that computer-based instruction will have in a specific setting. Effects of computer-based instruction may be generally positive but they are not totally predictable.

Level II analysis

The 97 studies can be classified by computer-use into 6 types:

1. *Tutoring*. The computer presents material, evaluates, responses, determines what to present next, and keeps records of progress. Computer uses classified as drill-and-practice and tutorial instruction in earlier taxonomies are covered by this term. The category is therefore similar to Taylor's (1980) category of computer-as-tutor.
2. *Managing*. The computer evaluates students either on-line or off-line, guides students to appropriate instructional resources, and keeps records.
3. *Simulation*. The computer generates data that meets student specifications and presents it numerically or graphically to illustrate relations in models of social or physical reality.

4. *Enrichment*. The computer provides relatively unstructured exercises of various types--games, simulations, tutoring, etc.--to enrich the classroom experience and stimulate and motivate students.

5. *Programming*. Students write short programs in such languages as Basic and Algol to solve mathematics problems. The expectation is that this experience in programming will have positive effects of students' problem-solving abilities and conceptual understanding of mathematics.

6. *Logo*. Students give the computer Logo instructions and observe the results on computer screens. From this experience students are expected to gain in ability to solve problems, plan, foresee consequences, etc.

Table 3 gives the means and standard deviations of effect sizes for studies in each of these categories. The table shows that effect sizes differ as a function of category of computer use. Results for three categories of computer use are especially noteworthy: results for computer tutoring, results for Logo programming, and other results.

Tutoring. The distribution of effect sizes for studies of tutoring is normal in shape (Figure 2). The average effect size is 0.38; the median effect is 0.36; and the standard deviation is 0.34. Comparing this distribution to the distribution for all studies yields some potentially important information. The mean of the distribution for tutoring studies is slightly higher than the mean of the

Table 3

Effect Sizes for 6 Categories Of Computer-Based Instruction

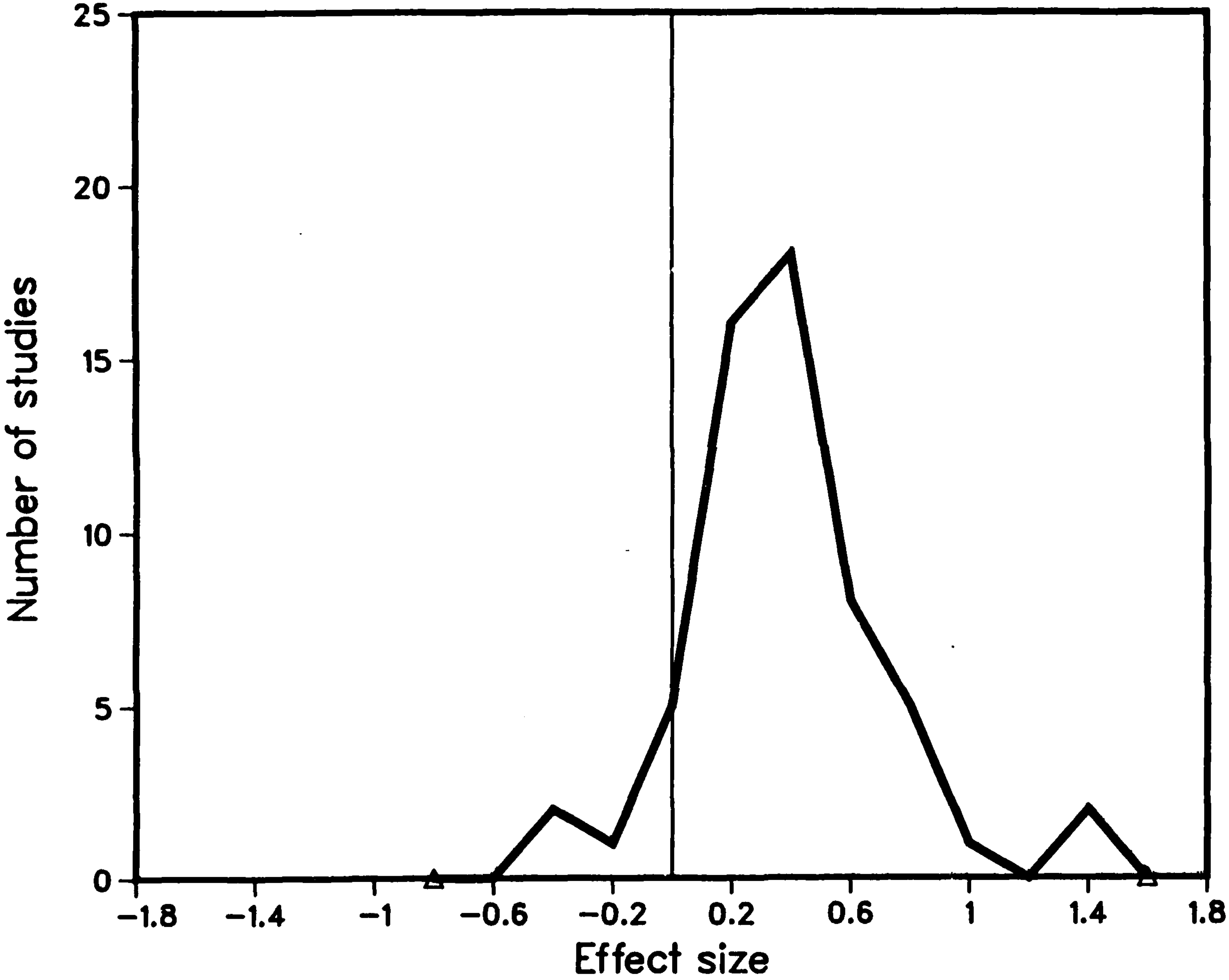
Application	Number of Studies	Effect Size	
		M	*SD*
Tutoring	58	0.38	0.34
Managing	10	0.14	0.28
Simulation	6	0.10	0.34
Enrichment	5	0.14	0.35
Programming	9	0.09	0.38
Logo	9	0.58	0.56

total distribution, and the standard deviation is slightly lower. Thus, if we know that a school system is employing its computers for tutoring, we would predict better than average results for a computer-based program, and we would also know that our predictions would be slightly more accurate than predictions that did not take program type into account.

Logo. The results for Logo programming are especially striking. The average effect size is high for the whole set of studies, but what is even more notable is the inconsistency in results. Some of the highest effect sizes in Table 2, for example, are associated with use of Logo. In a study by Rieber (1987), the scores on measures of problem-solving of children who learned with Logo were 1.5 standard deviations higher than the scores of children who were taught by conventional procedures. But not all studies produce such positive results. Most studies, in fact, report very small effects from Logo programs.

One difference between Logo studies reporting strong positive results and those reporting small effects is method of criterion measurement. In all the studies with strong positive results, the criterion test was individually administered. In all studies with weak results, criterion tests were group-administered. These facts raise questions about the meaning of the average effect for Logo studies. The strong positive results may have been produced by unusual evaluator rapport with Logo students during testing

Effects of Computer Tutoring On Examinations in 58 Studies



or even by unconscious bias in administering and recording responses of the Logo group. The fact is that Logo does not measure up on group tests, and these were the tests that were used in virtually all other studies of computer-based instruction. The case for strong benefits from Logo therefore seems unproved at this point.

Other uses of the computer. The record is also unimpressive for other approaches to computer-based instruction. Computer-managed instruction, for example, seldom produces significant positive gains in elementary and high schools. Its record of effectiveness seems similar to the record compiled by diagnostic and prescriptive systems that use only paper-and-pencil and printed materials in instructional delivery (Bangert-Drowns, Kulik, & Kulik, 1983). Programming in Basic or Algol does not usually have positive effects on student learning in mathematics courses. Learning of basic mathematical concepts, in fact, sometimes suffers with the introduction of computer programming into mathematics courses. Use of computer simulations in science courses also seems to have little effect on science learning in elementary and high school courses. More and better simulations may be needed to influence student examination performance.

Level III analysis

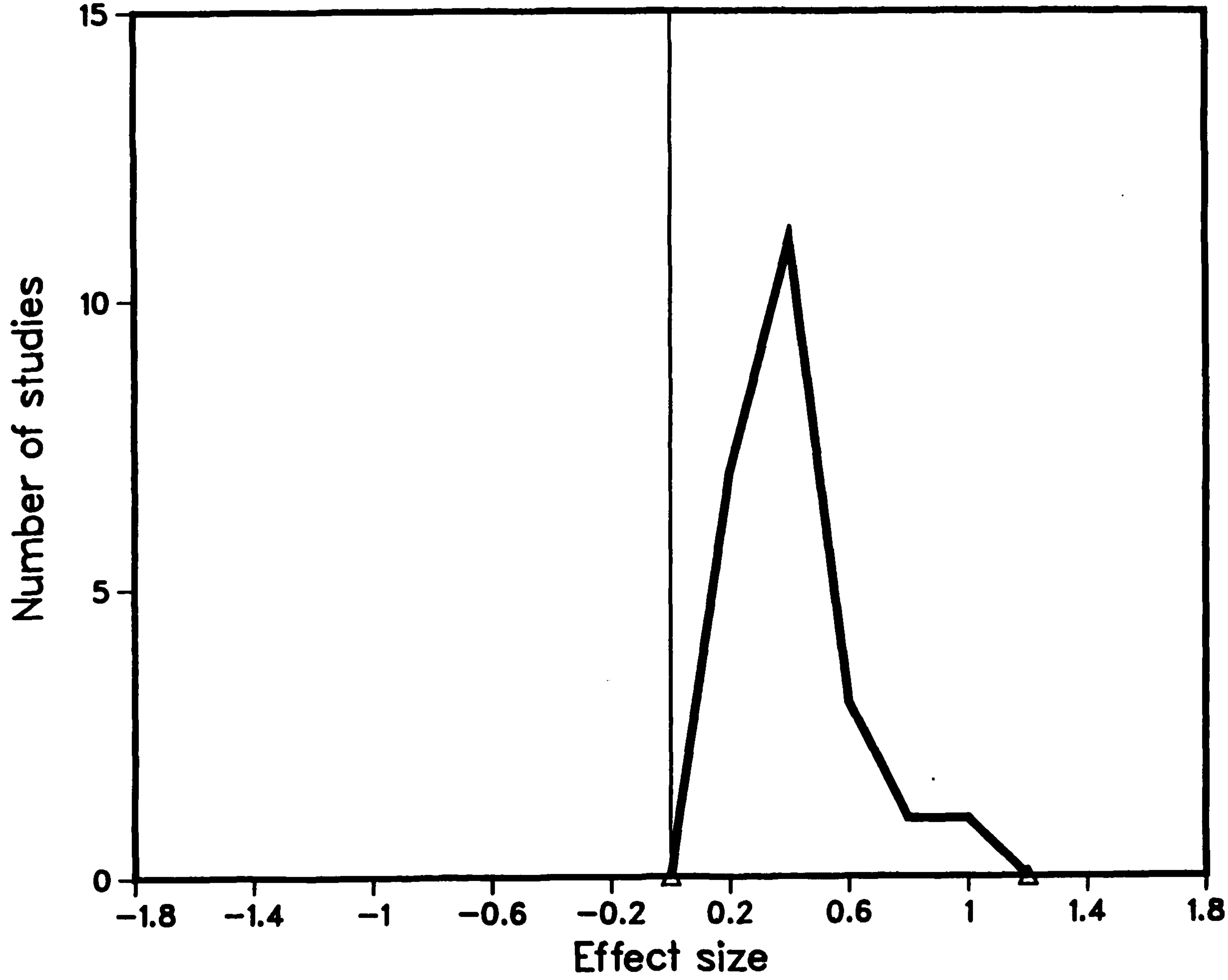
The Stanford-CCC program was evaluated in nearly two dozen controlled experiments during the past two decades. No other program of computer-based instruction has been the object of so much scrutiny. The accumulated studies on the Stanford-CCC programs are a unique resource in the evaluation of computer-based programs. We can use the studies to gauge the consistency of results from a Level III program. It is only natural to expect these results to be less variable than those we have already reviewed. But is the reduction in variability large or small?

The distribution of effect sizes from evaluations of the Stanford-CCC program is nearly normal in shape (Figure 3). The average effect size is 0.40; the median is 0.39; and the standard deviation is 0.23. The mean value is slightly higher and the dispersion is clearly smaller in this distribution than in the distributions in Figures 1 and 2. Thus, knowledge that a school is using the Stanford-CCC materials allows us to make clear and accurate predictions about what to expect. Gains of 1.4 years on a grade-equivalent scale are likely with a year-long program, whereas students who are conventionally instructed would gain only 1.0 year on the same scale. Gains of nearly 2.0 years are also quite possible, whereas gains of less than 1.0 years are highly unlikely.

Summary

The above analyses show that it is possible to make Level I generalizations about computer-based instruction. One such generalization is that computer-based instruction is usually effective instruction. But such a generalization is too gross. There are too many exceptions to the rule.

Effects of Stanford–CCC Programs On Examinations in 23 Studies



Our analyses also show that some types of computer-based instruction work better than others do. Statements about generic computer-based instruction are therefore of limited value. We need to go beyond generic conclusions and make statements about the effectiveness of specific types of computer-based instruction.

Ideally, reviewers would like to be able to form Level III generalizations about specific programs. If numerous evaluations were available on each specific program of computer-based instruction, then reviewers would be able to state with confidence how effective each approach was. But only one or two studies are available on most programs. Only the Stanford-CCC program has been evaluated frequently enough to warrant separate consideration in a Level III analysis. Based on the evaluation findings, we can state with confidence that this program produces positive results. It will probably be a long time before we can state with equal confidence that other specific programs work equally well.

Until that time we will probably have to content ourselves with Level II generalizations. If not so precise as Level III conclusions, they are nevertheless superior to the gross statements that result from a Level I analysis. Level II conclusions provide a better guide to both practitioners and researchers. They are an important step on the way to understanding the effects of specific programs.

Other Instructional Innovations

The most important Level II conclusion emerging from our analysis was on computer-based tutoring. Results from school programs that included such tutoring were better, on the average, than results from program without tutoring. But how important is the gain from computer tutoring? Is it as large as the gain from other innovative programs? Or do other innovations produce equally impressive or even better results?

To answer the question, I compared results from computer-based tutoring to results from other instructional innovations (Table 4). The results in Table 4 come from meta-analyses carried out at the University of Michigan (Bangert et al., 1983; Cohen, Kulik, & Kulik, 1982; C. Kulik, Kulik, & Bangert-Drowns, 1990; C. Kulik, Kulik, & Shwalb, 1982; J. Kulik & Kulik, 1984). Listed are eight instructional areas and the number of studies in each area. Also given is the average unadjusted effect size for each area. This is the average increase in examinations scores that is produced by use of the innovation, where the increase is measured in standard-deviation units.

Computer-based tutoring seems to be in the mid-range of instructional effectiveness. Higher effect sizes are associated with programs of accelerated instruction and mastery learning. Smaller effects are associated with the use of programmed texts and learning activity packages. Computer tutoring programs produce effects that are

Table 4

Unadjusted Effect Sizes for Computer Tutoring

And Other Innovations

Innovation	Number of studies	Unadjusted average effect size
Accelerated classes	13	0.88
Mastery learning	17	0.46
Peer & cross-age tutoring	52	0.40
Computer tutoring	58	0.38
Classes for gifted	29	0.37
Grouping	80	0.13
Learning packages	47	0.10
Programmed instruction	47	0.07

equivalent in size to those produced by programs of peer- and cross-age tutoring and by classes for gifted and talented students.

There is at least one major problem with these kinds of comparison. They ignore certain factors that affect evaluation results, including the types of examinations and experimental designs used in the studies. Because the average effect sizes listed in the table do not take into account evaluation styles in the different areas, the effect sizes given here are labeled *unadjusted effect sizes.*

A first important thing to notice about evaluation studies is their source. Findings in dissertations are almost always weaker than those reported in other sources, e.g., journal articles, books, and ERIC reports (Table 5). In our meta-analyses on precollege teaching, we did not find a single exception to this rule. It is not certain which set of findings--those from dissertations or those from other sources--are the more trustworthy. On the one hand, dissertation studies may be untrustworthy because they are the work of amateurs, whereas studies found in journals are more likely to be the work of professionals. On the other hand, journal studies may be untrustworthy because they have been carefully screened for statistical significance by editorial gatekeepers. Whatever the reason for the difference in dissertation and other results, it complicates comparison of studies from different areas, because in some

Table 5

Effect Sizes for Computer Tutoring and Other Innovations By Study Feature

Method	Source of document				Study duration				Criterion test			
	Dissertation		Other		Short		Long		Local		Standardized	
	N	*M*	*N*	*M*	*N*	*M*	*N*	*M*	*N*	*M*	*N*	*M*
Accelerated classes	3	0.72	10	0.93	0	----	13	0.88	0	----	13	0.88
Computer tutoring	24	0.31	33	0.45	11	0.40	84	0.39	20	0.36	37	0.40
Classes for the gifted	17	0.28	12	0.50	0	----	29	0.37	0	----	28	0.39
Grouping	28	0.04	52	0.17	0	----	80	0.13	6	-0.13	72	0.15
Learning packages	36	0.06	13	0.22	0	----	45	0.12	5	0.17	43	0.11
Mastery learning	4	0.40	13	0.49	3	0.46	14	0.47	12	0.63	5	0.08
Programmed instruction	27	0.02	20	0.18	15	0.19	32	0.03	20	0.17	27	0.02
Peer & cross-age tutoring	30	0.27	22	0.58	6	0.95	44	0.34	12	0.84	40	0.27

areas most studies are carried out by graduate students as dissertation research, whereas in other areas most studies are found in journals.

A second factor that seems to influence the outcomes of evaluations at the precollege level is the type of examination used as a criterion measure. Findings on evaluator-designed local measures are usually clearer than findings on standardized measures of school achievement (Table 5). It may be that evaluator-designed measures are unconsciously biased toward the experimental treatments, or it may be that standardized tests are too global to use to evaluate specific curricula. Whatever the case is, it seems to me unfair to compare effects from different areas when evaluation studies in some areas rely heavily on local tests and evaluation studies in other areas rely largely on standardized tests.

A third factor that affects the outcomes of precollege studies is their duration. Short studies--where short is defined as a duration of less than four weeks--often produce stronger findings than do long studies (Table 5). Again, it is hard to say which sort of study we should trust. Short studies may be better controlled, but long studies are certainly more ecologically valid. The problem is that short studies are common in some evaluation areas and rare in others.

Table 6

Adjusted Effect Sizes for Computer Tutoring

And Other Innovations

Innovation	Number of studies	Adjusted average effect size
Accelerated classes	13	0.93
Classes for gifted	29	0.50
Computer tutoring	58	0.48
Peer & cross-age tutoring	52	0.38
Grouping	80	0.19
Learning packages	47	0.19
Mastery learning	17	0.10
Programmed instruction	47	0.07

Table 6 also shows adjusted effect sizes for the eight areas. These are the average effect sizes that we would expect if all the studies in each area were of the same type. I used multiple regression techniques to make the adjustments. The adjusted effect sizes are the best estimates of effects for studies that (a) are reported in journal articles or technical reports, (b) use standardized tests as the criterion measures, and (c) are at least one month in duration.

I think that these adjusted results are clearer than the unadjusted results. The table suggests to me that innovations that make the biggest difference involve curricular change for high-achieving individuals. Schools can dramatically improve the achievement of their high-aptitude learners by giving them school programs that provide greater challenge. The next most potent innovations involve individual tutoring by computers or by other students. At this point, computer tutoring seems to be slightly more effective than peer- and cross-age tutoring. Instructional technologies that rely on paper-and-pencil are at the bottom of the scale of effectiveness.

Conclusions

Meta-analysts have demonstrated repeatedly that programs of computer-based instruction usually have positive effects on student learning. This conclusion has emerged from too many separate meta-analyses to be considered controversial. Nonetheless, results are not the same in every study of computer-based instruction. No meta-analyst has reported that all types of computer-based instruction increase student achievement in all types of settings. Study results are not that consistent, nor would we want them to be. Computer-based instruction is a loose category of innovations. It covers some practices that usually work and other programs that have little to offer.

Breaking studies of computer-based instruction into conventional categories clarifies the evaluation results. One kind of computer application that usually produces positive results in elementary and high school classes is computer tutoring. Students usually learn more in classes that include computer tutoring. On the other hand, precollege results are unimpressive for several other computer applications: managing, simulations, enrichment, and programming. Results of Logo evaluations are variable. Logo evaluations that measure gains on individual tests report highly positive results. Logo evaluations that employ group tests report indifferent results.

The overall findings on computer tutoring compare favorably with findings on other innovations. Few innovations in precollege teaching have effects as large as those of computer tutorials. Effects are especially large and consistent in well-designed programs such as the Stanford-CCC program. Programs of curricular change that provide more challenge for high-aptitude students may have produced more dramatic effects in evaluation studies, but such programs affect only a limited part of the school population. The effects of computer tutoring are as great as those of peer- and cross-age tutoring, and they are clearly greater than the gains produced by instructional technologies that rely on print materials.

References

Adams, M., Ingebo, G., Leitner, D., Miller, S., Mollanen, C., & Peters, L. (1983). *Computer use in the Portland public schools, an evaluation: Computer-based instruction, computer literacy programs, computer management*. Portland Public Schools, OR. Evaluation Department. (ERIC Document Reproduction Service No. ED 240 158)

Alberta Department of Education. (1983). *Using the computer to teach language arts*. Planning Services Branch, Edmonton Public Schools, Alberta. (ERIC Document Reproduction Service No. ED 252 861)

Al-Hareky, S. M. (1984). A study of the effectiveness of modern educational technology on the mathematics performance of elementary students in Saudi Arabia. *Dissertation Abstracts International*, *45*, 734A. (University Microfilms No. 84-19,010)

Anderson, T. E. (1984). Traditional method versus computer-aided instruction in teaching keyboarding skills to elementary school-age children. *Dissertation Abstracts International*, *45*, 388A. (University Microfilms No. 84-12,292)

Ash, R. C. (1986). Analysis of the cost-effectiveness of a computer-assisted instruction program. *Dissertation Abstract International*, *46*, 2137. (University Microfilms No. 85-23,784)

Atkinson, R. C. (1969). Computerized instruction and the learning process. In R. C. Atkinson & H. A. Wilson (Eds.), *Computer-assisted Instruction: A book of readings* (pp. 143-165). New York: Academic Press.

Bangert, R. L., Kulik, J. A., & Kulik, C.-L. C. (1983). Individualized systems of instruction in secondary schools. *Review of Educational Research, 53*, 143-158.

Bangert-Drowns, R. L., Kulik, J. A., & Kulik, C.-L. C. (1985). Effectiveness of computer-based education in secondary schools. *Journal of Computer-Based Instruction, 12*, 59-68.

Beck, D. & Chamblain, E. (1983). *Language development component, secondary developmental reading program. Final evaluation report*. Columbus Public Schools, Department of Evaluation Services, Columbus, OH. (ERIC Document Reproduction Service No. ED 249 252)

Bostrom, K., Cole, A. J., Hartley, J. R., Lovell, K., & Tait, R. (1982). *An evaluative study of the effects and effectiveness of microcomputer based teaching in schools*. Social Science Research Council, London, England.

Boyd, A. L. (1973). Computer aided mathematics instruction for low-achieving students. *Dissertation Abstracts International, 33*, 553A. (University Microfilms No. 73-17,131)

Broderick, W. R., & others. (1973). *Off-line computer aided learning project: The development of educational material. The second report.* London: Social Science Research Council. (ERIC Document Reproduction Service No. ED 129 279)

Burns, P. K., & Bozeman, W. C. (1981). Computer-assisted instruction and mathematics achievement: Is there a relationship? *Educational Technology, 21*, 32-39.

Chamberlain, E., & Beck, D., & Johnson, J. (1983). *Language development component, compensatory language experiences and reading program.* Final Evaluation Report. Columbus Public Schools, OH. Department of Evaluation Services. (ERIC Document Reproduction Service No. ED 249 247)

Chiang, A., Stauffer, C., & Cannara, A. (1978). *Demonstration of the use of computer-assisted instruction with handicapped children, final report.* (Report No. 446-AH-60076A). Arlington, VA: RMC Research Corporation. (ERIC Document Reproduction Service No. ED 166 913)

Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*, 309-318.

Coffman, W. E. & Olsen, S. A. (1980). *The first two years of PLAN: An evaluation of program impact.* Iowa City, Iowa: Iowa Testing Programs. (ERIC Document Reproduction Service No. ED 190 674)

Cohen, P. A., Kulik, J. A., & Kulik, C.-L. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal, 19*, 237-248.

Cole, W. L. (1971). The evaluation of a one-semester senior high school mathematics course designed for acquiring basic mathematical skills using computer-assisted instruction. *Dissertation Abstracts International, 32*, 2399A. (University Microfilms No. 71-29,729)

Confer, R. W. (1971). The effect of one style of computer assisted instruction on the achievement of students who are repeating general mathematics. *Dissertation Abstracts International, 32*, 1741A. (University Microfilms No. 72-26,160)

Coomes, P. A. (1986). The effects of computer assisted instruction on the development of reading and language skills. *Dissertation Abstracts International, 46*, 3302A. (University Microfilms No. 85-27,359)

Cooperman, K. S. (1985). An experimental study to compare the effectiveness of a regular classroom reading program to a regular classroom reading program with a computer-assisted instruction program in reading comprehension skills in grades two through four. *Dissertation Abstracts International, 46*, 1234A. (University Microfilms No. 85-15,764)

Cranford, H. R. (1976). A study of the effects of computer assisted instruction in mathematics on the achievement and attitude of pupils in grades five and six in a rural setting. *Dissertation Abstracts International*, *37*, 5660A. (University Microfilms No. 77-5932)

Crawford, A. N. (1970). A pilot study of computer assisted drill and practice in seventh grade remedial mathematics. *California Journal of Educational Research*, *21*, 170-174.

Davies, T. P. (1972). An evaluation of computer assisted instruction using a drill and practice program. *Dissertation Abstracts International*, *32*, 6970B. (University Microfilms No. 72-18,627)

Degelman, D., Free, J. U., Scarlato, M., Blackburn, J. M., & Golden, J. (1986). Concept learning in preschool children: Effects of a short-term Logo experience. *Journal of Educational Computing Research*, *2*, 199-205.

Delon, F. G. (1970). A field test of computer assisted instruction in first grade mathematics. *Educational Leadership*, *28*, 170-180.

Diamond, J. J. (1969). *A report on project GROW: Philadelphia's experimental program in computer assisted instruction*. Philadelphia: Philadelphia School District. (ERIC Document Reproduction Service No. ED 035 272)

Diamond, R. C. (1986). Traditional method versus computer-game instruction in teaching keyboarding skills to kindergarten children. *Dissertation Abstracts International*, *46*, 2555A. (University Microfilms No. 85-26,333)

Dunn, A. (Ed.). (1974). *Computer-assisted instruction program. A three-year report covering July 1, 1971, through June 30, 1974*. Rockville, MD: Montgomery County Public Schools. (ERIC Document Reproduction Service No. ED 100 361)

Durward, M. (1973). *Computer assisted instruction in arithmetic at South Hill Elementary School*. British Columbia: Vancouver Board of School Trustees. (ERIC Document Reproduction Service No. ED 088 915)

Easterling, B. A. (1982). The effects of computer assisted instruction as a supplement to classroom instruction in reading comprehension and arithmetic. *Dissertation Abstract International*, *43*, 2231 A. (University Microfilms No. 82-23,032)

Fejfar, F. L. (1969). ISU lab school fourth graders learn through CAI. *Contemporary Education*, *40*, 296-297.

Feldhausen, M. W. (1986). The effects of computer review assistance modules (CRAM) on student achievement in United States history. *Dissertation Abstracts International*, *47*, 68A. (University Microfilms No. 86-06,961)

Ferrell, B. G. (April, 1985). *Computer immersion project: Evaluating the impact of computers on learning*. Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL. (ERIC Document Reproduction Service No. ED 259 951)

Fisher, M. E. (1973). A comparative study of achievement in the concepts of fundamentals of geometry taught by computer managed individualized behavioral objective instructional units versus lecture-demonstration methods of instruction. *Dissertation Abstracts International*, *34*, 2161A. (University Microfilms No. 73-25,330)

Fletcher, J. D. & Atkinson, R. C. (1972). Evaluation of the Stanford CAI program in initial reading. *Journal of Educational Psychology*. *63*, 597-602.

Foster, T. E. (1973). The effect of computer programming experiences on student problem solving behaviors in eighth grade mathematics. *Dissertation Abstracts International*, *33*, 4239A. (University Microfilms No. 72-31,527)

Gershman, J., & Sakamoto, E. (1981). Computer-assisted remediation and evaluation: A CAI project for Ontario secondary schools. *Educational Technology*, *21*, 40-43.

Glass, G. V., McGaw, B., & Smith, M. L. (1981). *Meta-analysis in social research*. Beverly Hills, Calif.: Sage Publications.

Grocke, M. (1982). *Interactive Development of Reading Skills in an Educational Clinic*. Australia, paper presented at the Annual National Conference of the Australian Group for the Scientific Study of Mental Deficiency. (ERIC Document Reproduction Service No. ED 223 993)

Haberman, E. L. (1977). Effectiveness of computer assisted instruction with socially/emotionally disturbed children. *Dissertation Abstracts International*, *38*, 1998A. (University Microfilms No. 77-21,221)

Hartley, S. S. (1978). Meta-analysis of the effects of individually paced instruction in mathematics. *Dissertation Abstracts International*, *38*(7-A), 4003. (University Microfilms No. 77-29,926)

Hatfield, L. L. (1970). Computer-assisted mathematics: An investigation of the effectiveness of the computer used as a tool to learn mathematics. *Dissertation Abstracts International*, *30*, 4329A. (University Microfilms No. 70-5569)

Hedges, L. V., & Olkin, I. (1985). *Statistical methods for meta-analysis*. Orlando, FL: Academic Press.

Henderson, R. W., Landesman, E. M., & Kachuk, I. (April, 1983). *Effects of interactive video computer instruction on the performance of underachieving students in mathematics*. Paper presented at the annual meeting of the American Educational Research Association, Montreal,Canada. (ERIC Document Reproduction Service No. ED 233 690)

Horner, C. M. (1984). The effects of Logo on problem solving, locus of control, attitudes toward mathematics, and angle recognition in learning disabled children. *Dissertation Abstracts International, 45*, 1716A. (University Microfilms No. 84-19,865)

Horner, C. M., & Maddux, C. D. (1985). The effect of Logo on attributions toward success. *Computers in the Schools, 2*, 45-54.

Horton, J., & Ryba, K. (1986). Assessing learning with Logo: A pilot study. *Computing Teacher, 14*, 24-28.

Hughes, W. R. (1974). A study of the use of computer simulated experiments in the physics classroom. *Dissertation Abstracts International, 34*, 4910A. (University Microfilms No. 74-3205)

Hunter, J. E., Schmidt, F. L., & Jackson, G. B. (1982). *leta-analysis: Cumulating research findings across studies*. Beverly Hills, CA: Sage.

Jamison, D., Fletcher, J. D., Suppes, P., & Atkinson, R. C. (1976). Cost and performance of computer assisted instruction for education of disadvantaged children. In J. Froomkin & R. Wadner (Eds.), *Education as an Industry*. New York: Columbia University Press.

Jhin, K. R. (1971). A statistical comparison of the effectiveness of non-tutorial computer aided and conventional teaching of algebra. *Dissertation Abstracts International, 32*, 5734A. (University Microfilms No. 72-11278)

Johnson, R. E. (1971). The effect of activity oriented lessons on the achievement and attitudes of seventh grade students in mathematics. *Dissertation Abstracts International, 32*, 304A. (University Microfilms No. 71-18,720)

Jones, J. E. (1974). Computer-simulated experiments in high school physics and chemistry. *Dissertation Abstracts International, 34*, 6903A. (University Microfilms No. 73-3897)

Katz, S. (1971). A comparison of the effects of two computer augmented methods of instruction with traditional methods upon achievement or Algebra Two students in a comprehensive high school. *Dissertation Abstracts International, 32*, 1188A. (University Microfilms No. 71-19,986)

Kieren, T. E. (1969). The computer as a teaching aid for eleventh grade mathematics: A comparison study. *Dissertation Abstracts, 29*, 3526A. (University Microfilms No. 68-17,690)

Krull, S. M. (1980). An analysis of the effects of learning to program on student math performance and attitude toward school. *Dissertation Abstract International*, *40*, 5711A. (University Microfilm No. 80-10,736)

Kulik, C.-L. C., & Kulik, J. A. (1986). Effectiveness of computer-based education in colleges. *AEDS Journal*, *19*, 81-108.

Kulik, C.-L. C., & Kulik, J. A. (In press). Effectiveness of computer-based instruction: An updated analysis. *Computers in Human Behavior*.

Kulik, C.-L. C., Kulik, J. A., & Bangert-Drowns, R. L. (1990). Effectiveness of mastery learning programs: A meta-analysis. *Review of Educational Research*, *60*, 265-299.

Kulik, C.-L. C., Kulik, J. A., & Shwalb, B. J. (1986). The effectiveness of computer-based adult education: A meta-analysis. *Journal of Educational Computing Research*, *2*, 235-252.

Kulik, C.-L. C., Shwalb, B. J., & Kulik, J. A. (1982). Programmed instruction in secondary education: A meta-analysis of findings. *Journal of Educational Research*, *75*, 133-138.

Kulik, J. A., & Kulik, C.-L. C. (1984). Effects of accelerated instruction on students. *Review of Educational Research*, *54*, 409-426.

Kulik, J. A., & Kulik, C.-L. C. (1989). Meta-analysis in education. *International Journal of Educational Research*, *13*, 221-340.

Kulik, J. A., & Kulik, C.-L. C. (1991). Ability grouping and gifted students. In N. Colangelo & G. Davis (Eds.), *Handbook of gifted education*. MA: Allyn & Bacon. (Pp. 178-196).

Kulik, J. A., Kulik, C.-L. C., & Bangert-Drowns, R. L. (1985a). Effectiveness of computer-based education in elementary schools. *Computers in Human Behavior*, *1*, 59-74.

Lally, M. (1981). Computer-assisted teaching of sight-word recognition for mentally retarded school children. *American Journal of Mental Deficiency*, *85*, 383-388.

Lang, C. R. (1976). Computer graphic simulations in high school physics. *Dissertation Abstracts International*, *37*, 903A. (University Microfilms No. 76-17,116)

Larrea-Peterson, M. (1986). A comparison of reading and mathematics achievement and cost effectiveness: Traditional and computerized 'Prescription Learning' methods; Grades 2-6, Salt Lake City School District, 1983-1984. *Dissertation Abstracts International*, *47*, 39A. (University Microfilms No. 86-04,742)

Levy, M. H. (1985). An evaluation of computer assisted instruction upon the achievement of fifth grade students as measured by standardized tests. *Dissertation Abstracts International*, *46*, 860A. (University Microfilms No. 85-13,059)

Litman, G. (1977). Relation between computer-assisted instruction and reading achievement among fourth, fifth, and sixth grade students. *Dissertation Abstracts International*, *38*, 2003A. (University Microfilms No. 77-20,883)

Lunetta, V. N. (1972). The design and evaluation of a series of computer simulated experiments for use in high school physics. *Dissertation Abstracts International*, *33*, 2785A. (University Microfilms No. 72-32,153)

MacLean, R. F. (1974). A comparison of three methods of presenting instruction in introductory multiplication to elementary school children (total computer, partial computer, and non-computer. *Dissertation Abstracts International*, *35*, 1430. (University Microfilms No. 74-19,759)

Mandelbaum, J. (1974). A study of the effects, on achievement and attitude, of the use of the computer as a problem solving tool with low performing tenth grade students. *Dissertation Abstracts International*, *34*, 3700A. (University Microfilms No. 74-1809)

McEwen, N., & Robinson, A. (1976). *Computer-assisted instruction in secondary school French. Final Report.* Edmonton: Alberta University. (ERIC Document Reproduction Service No. ED 150 846)

Melnik, L. (1986). An investigation of two methods for improving problem solving performance of fifth grade students. *Dissertation Abstracts International, 47*, 405. (University Microfilm No. 86-05,549)

Mendelsohn, M. (1972). CAI in New York City: The slow learner in mathematics. *National Council of Teachers of Mathematics Yearbook*, 355-364.

Metric Associates. (1981). *Evaluation of the computer-assisted instruction Title I Project, 1980-81.* Chelmsford, MA: Metric Associates. (ERIC Document Reproduction Service No. ED 233 122)

Miller, S. W. (1984). A comparison of computer-assisted instruction with prescription learning and the traditional "pull-out" program used in supplementing of mathematics basic skills to Chapter I (Title I) students. *Dissertation Abstracts International, 44*, 2397A. (University Microfilms No. 83-26,453)

Mitzel, H. (1971). *A commonwealth consortium to develop, implement, and evaluate a pilot program of computer assisted instruction for urban high schools. Final Report.* Pittsburgh: University of Pittsburgh. (ERIC Document Reproduction Service No. ED 059 604)

Morgan, C. E., Sangston, B. J., & Pokras, R. (1977). *Evaluation of Computer-Assisted Instruction*, 1975-76. Rockville, MD: Montgomery County Public Schools. (ERIC Document Reproduction Service No. ED 139 655)

Mravetz, P. J. (1980). The effects of computer-assisted instruction on student self concept, locus of control, level of aspiration, and reading achievement. *Dissertation Abstracts International*, *41*, 994A.

Nabors, D. G. (1974). A comparative study of academic achievement and problem-solving abilities of black pupils at the intermediate level in computer-supported instruction and self-contained instructional programs. *Dissertation Abstracts International*, *36*, 3241A. (University Microfilms No. 75-26,294)

Niemiec, R. P., & Walberg, H. J. (1985). Computers and achievement in the elementary schools. *Journal of Educational Computing Research*, *1*, 435-440.

Niemiec, R., & Walberg, H. J. (1987). Comparative effects of computer-assisted instruction: A synthesis of reviews. *Journal of Educational Computing Research*, *3*, 19-37.

O'Connell, W. B., Jr. (1973). An investigation of the value of exposing slow-learner ninth year mathematics pupils to a relatively short computer experience. *Dissertation Abstracts International*, *34*, 124A. (University Microfilms No. 73-14,846)

Odom, M. L. N. (1984). The effect of learning the computer programming language Logo on fifth and six grade student's skill of analysis, synthesis and evaluation. *Dissertation Abstracts International*. (University Microfilms No. 84-26,197)

Pachter, S. N. (1979). A computer assisted tutorial module for teaching the factoring of second degree polynomials to regents level ninth year mathematics students. *Dissertation Abstracts International*, *40*, 1843A. (University Microfilms No. 79-23,610)

Palmer, H. (1973). *Three evaluation reports of computer-assisted instruction in drill and practice mathematics*. CA: Los Angeles County Superintendent of Schools. (ERIC Document Reproduction Service No. ED 087 422)

Porinchak, P. M. (1984). Computer-assisted instruction in secondary school reading: Interaction of cognitive and affective factors. *Dissertation Abstracts International*, *45*, 478A. (University Microfilms No. 84-10,506)

Prince, J. D. (1969). *A practitioner's report results of two years of computer-assisted instruction in drill and practice mathematics*. McComb, Mississippi, McComb Schools. (ERIC Document Reproduction Service No. ED 032 769)

Ragosta, M. (1983). Computer-assisted instruction and compensatory education: a longitudinal analysis. *Machine-Mediated Learning*, *1*, 97-127.

Reimer, G. (1985). The effects of Logo computer programming experience on readiness of first grade, creativity, and self concept. *AEDS Monitor*, *23*, 8-12.

Rieber, L. P. (1987). Logo and its promise: A research report. *Educational Technology*, 12-16.

Roberts, A. S. (1982). The effects of split-day scheduling and computer-managed instruction on the reading achievement of intermediate students. *Dissertation Abstracts International*, *43*, 1482A. (University Microfilms No. 82-23,584)

Rodefer, J. C. (1986). Teaching higher level thinking skills through Logo. *Dissertation Abstracts International*. (University Microfilms No. 86-803)

Ronan, F. D. (1971). A study of the effectiveness of a computer when used as a teaching and learning tool in high school mathematics. *Dissertation Abstracts International*, *32*, 1264A. (University Microfilms No. 71-23,861)

Rosenthal, R. (1984). *Meta-analytic procedures for social research*. Beverly Hills, CA: Sage.

Schmidt, M., Weinstein, T., Niemiec, R., & Walberg, H. J. (1985). *Computer-assisted instruction with exceptional children: A meta-analysis of research findings*. Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Shaw, D. G. (1986). Effects of learning to program a computer in basic Logo on problem-solving abilities. *AEDS Journal*, 176-189.

Smith, E. S. (1980). The effect of computer-assisted instruction on academic achievement, school daily attendance, and school library usage at Margaret Murray Washington Career Center. *Dissertation Abstracts International*, *41*, 2431A. (University Microfilms No. 80-26,342)

Sperry, J. W. (1977). Computer simulations and critical thinking in high school biology. *Dissertation Abstracts International*, *37*, 5370A. (University Microfilms No. 77-4669)

Staniskis, C. C. (1977). A comparison of student content achievement in biology between computer managed instructional and non-computer managed instructional biology courses. *Dissertation Abstracts International*, *37*, 7665A. (University Microfilms No. 77-13,528)

Summerlin, L., & Gardner, M. (1973). A study of tutorial-type computer assisted instruction in high school chemistry. *Journal of Research in Science Teaching*, *10*, 75-82.

Suppes, P. & Morningstar, M. (1969). Computer-assisted instruction. *Science*, *166*, 343-350.

Suppes, P., & Morningstar, M. (1970). Technological innovations: Computer-assisted instruction and compensatory education. In F. F. Korten, S. W. Cook, &

J. I. Lacey (Eds.), *Psychology and the problems of society* (pp. 221-236). Washington, DC: American Psychological Association.

Taylor, R. P. (Ed.) (1980). *The computer in the school: Tutor, tool, tutee.* New York: The Teachers College Press.

Thompson, B. B. (1973). Effect of computer-assisted instruction on the language arts achievement of elementary school children. *Dissertation Abstracts International, 33*, 4077-4078A. (University Microfilms No. 73-3574)

Todd, W. E. (1986). Effects of computer-assisted instruction on attitudes and achievements of fourth grade students in reading and mathematics. *Dissertation Abstracts International, 46*, 3249. (University Microfilm No. 85-27,393)

Turner, L. G. (1986). An evaluation of the effects of paired learning in a mathematics computer-assisted-instruction program. *Dissertation Abstracts International, 46*, 3641A. (University Microfilms No. 86-02,860)

Vincent, A. T. (1977). The effects of supplementary computer-assisted instruction on the mathematics achievement and attitude toward mathematics of EMR high school students. *Dissertation Abstracts International, 39*, 736A.

Wainwright, C. L. (1985). The effectiveness of a computer-assisted instruction package in the supplementing teaching of selected concepts in high school chemistry: Writing formulas and balancing chemical equations. *Dissertation Abstracts International*, *45*, 2473A. (University Microfilms No. 84-24,757)

Warner, T. (1979). *The Jackson elementary school computer based education mathematics project*. Akron, OH: The University of Akron, Computer Based Education Center.

Way, J. W. (1984). *Evaluation of computer assisted instruction*. Kansas City School District, MO. (ERIC Document Reproduction Service No. ED 257 840)

Willett, J. B., Yamashita, J. J., & Anderson, R. D. (1983). A meta-analysis of instructional systems applied in science teaching. *Journal of Research in Science Teaching*, *20*, 405-417. f

Wilson, H. A. & Fitzgibbon, N. H. (1970). Practice and perfection: A preliminary analysis of achievement data from the CAI elementary English program. *Elementary English*, *47*, 576-579.

Wilson, P. M. (1982, July). *Computer-based education and proficiency testing: A model of cost-effectiveness*. Paper presented at the SALT Conference on Training Effectiveness and Evaluation.

Wolf, F. M. (1986). *Meta-analysis: Quantitative methods for research synthesis*. Beverly Hills, CA: Sage.

ASSESSMENT OF SOFTWARE STRATEGIES

Assessment of Explanation Systems

Johanna D. Moore
University of Pittsburgh
Department of Computer Science
and
Learning Research and Development Center
Pittsburgh, PA 15260
jmoore@cs.pitt.edu

Abstract

The need for explanation capabilities in intelligent systems, and especially expert systems, has been widely recognized and most commercially available expert system building tools provide at least rudimentary explanation capabilities. Yet, few rigorous and thorough evaluations have been attempted. This is in part due to the fact that an explanation system is not stand-alone, but occurs as a component of a larger, intelligent system and the quality of a system's explanations depends on factors outside the control of the explanation component. Criteria and methods for evaluating explanation systems are only now beginning to emerge. In this paper, I describe the goals of explanation systems and discuss three methods for determining if those goals are being achieved: assessing the impact of the system on the user's satisfaction or performance, comparing human-machine explanation interactions to human-human advisory interactions, and assessing a system according to a set of evaluation criteria. The first two methods have been employed and the results of case studies using these methods are discussed. The third method relies on developing appropriate evaluation criteria and a set of recently proposed criteria is presented. These criteria include metrics such as the cost of providing a system with explanation capabilities and the benefits gained from making this investment. Current research indicates that providing a system with explanation capabilities requires substantially more effort during system development, but that these costs are more than recovered during the software lifecycle because the kinds of architectures that support explanation also facilitate knowledge acquisition and system maintenance. New developments in explainable system architectures are described.

1 Explanation Systems: Definition and Goals

As the name suggests, an explanation system is a system that presents its users with explanatory information in response to user requests. An explanation system is not stand-alone, but rather a component of a knowledge-based system such as an expert, advice-giving, or intelligent tutoring system. The explanation component of such a system is intended to elucidate the intelligent system's domain knowledge and behavior by producing explanations such as definitions of terms, justification of results, and comparisons of alternate methods for solving problems or achieving goals.

The critical need for explanation has been voiced by expert system builders and the intended user community alike. In a study by Teach and Shortliffe (1984) in which physicians were asked to rank 15 capabilities of computer-based consultation systems in order of importance, they rated the ability "to explain their diagnostic and treatment decisions to physician users" as the most essential of the capabilities surveyed. Third on the list was the ability to "display an understanding of their own medical knowledge." The importance of explanation capabilities to these users is underscored by the fact that the capability to "never make an incorrect diagnosis" was ranked 14th out of 15 capabilities surveyed! This suggests that explanation capabilities are not only desirable, but crucial to the success of knowledge-based systems.

Researchers have designed explanation systems with a variety of goals in mind. For end users, these goals include:

- **information:** Explanation systems offer access to the considerable knowledge available in an intelligent system.
- **education:** Explanations can be used to educate users about the problem domain.
- **acceptance:** The advice of intelligent systems will not be accepted unless users believe they can depend on the accuracy of that advice. Systems that can display their domain knowledge and justify the methods used in reaching conclusions are considered more acceptable to the user community [Teach and Shortliffe, 1984].
- **assess appropriateness of system for task:** The scope of intelligent systems is typically narrow. Swartout (1983) has pointed out that an explanation facility can help a user discover when a system is being pushed beyond the bounds of its expertise by presenting the methods being employed and rationales for employing them in a given situation.

In addition to the benefits offered to end-users of systems, an explanation facility can also be useful for system developers. An additional goal of explanation system is to:

- **aid in debugging and maintaining system:** Because a textual or graphical explanation of a system's knowledge or behavior offers a viewpoint different from the code that implements it, system developers find that such explanations can aid in locating bugs and also offer a form of documentation that is helpful in maintenance.

2 Difficulties in evaluating explanation systems

Research in the area of expert system explanation has led to the understanding that explanation capabilities cannot be added to a system as an afterthought. The capability to support explanation imposes stringent requirements on the design of an expert system and it can be difficult, if not impossible, to endow a system with the capability to produce adequate explanations unless those requirements are taken into consideration when designing the system [Clancey, 1983b, Swartout, 1983]. Retrofitting an existing intelligent system with explanation capabilities inevitably means redesigning it.

The range of user questions an explanation facility is able to handle and the sophistication of the responses it can produce depends on two knowledge sources: (1) knowledge about the domain and how to solve problems in that domain as represented in the intelligent system's knowledge base, e.g., definitions of terminology, justifications for actions; and (2) knowledge about how to construct an adequate response to a user's query in some communication medium, e.g., natural language and/or graphics. The latter includes methods for interpreting the user's questions and strategies for choosing information relevant to include in answers to different types of questions.

In the expert systems area, researchers have determined that many of the inadequacies noted in the explanations stem from limitations in the systems' knowledge bases. Until recently, expert system architects have concentrated their efforts on the problem-solving needs of systems. They designed knowledge bases and control mechanisms best suited to performing an expert task. Because of this, the explanation capabilities of these systems were limited to producing procedural descriptions of *how* a given problem is solved. Knowledge needed to justify the system's actions, explain general problem solving strategies, or define the terminology used by the system was simply *not* represented and therefore could not be included in explanations [Clancey, 1983b, Swartout, 1983]. Deficiencies in the knowledge representation leads to other types of inadequacies, rendering systems inflexible, insensitive, and unresponsive to users' needs. To provide different explanations to different users in different situations or in response to requests for elaboration, the system must have a rich representation of its knowledge, including abstract strategic knowledge as well as detailed knowledge, a rich terminological base, and causal knowledge of the domain.

The fact that the explanation capabilities of a system are tightly coupled to the knowledge base and reasoning component of that system leads to a fundamental problem for those interested in the assessment of explanation systems. An explanation facility cannot be evaluated on its own, it must be evaluated in the context of the larger intelligent system of which it is a part. It would be unreasonable to fault the explanation facility for its inability to respond to users' requests for rule justifications, when the knowledge need to justify rules in not present in the system's knowledge base. Thus, evaluating an explanation facility is a complex task. It inevitably involves assessing the entire intelligent system, dissecting the system into components so that blame may fairly be assigned to the offending component(s), and defining the types of knowledge needed to produce explanations, all in order to gain an understanding of what limitations could stem from the knowledge base and what limitations stem from the explanation component, itself.

In the sections that follow, I describe three methods that could be used to evaluate explanation systems. The first two methods have been employed and I describe two specific cases in detail. As we will see, these case studies had the goal of identifying specific problems with existing explanation systems and uncovering the limitations behind the perceived inadequacies. Such studies led researchers to an understanding of the requirements that the need to provide explanations places on the knowledge representation and reasoning processes of an intelligent system. In this paper, I describe the limitations identified in the case studies as well as the general principles that came from in-depth analyses of these limitations. Finally, I discuss current research efforts in explanation technologies which are attempting to provide applications builders with an explainable intelligent system framework by capturing the knowledge needed to support explanation and structuring that knowledge so that it is available to the explanation facility. While such frameworks are still exploratory, and the feasibility of providing domain-independent reasoning and explanation facilities remains to be demonstrated, many promising results are emerging from this work. One intriguing possibility from the perspective of evaluation is that the requirements posed by an explainable intelligent system framework may in fact suggest a way to evaluate *whether* a system can readily accept an explanation module.

3 Methods For Evaluation

Given the nature of explanation systems and their tight coupling to other components of intelligent systems, I believe there are three possible methods for assessing explanation capabilities.

3.1 Assess impact on user

The purpose of an explanation component is to facilitate the user's access to the information and knowledge stored in the intelligent system. Thus one way to assess an explanation component, is to assess the impact of the explanation component on the user's behavior or satisfaction with the system. This can be done using direct methods such as interviewing users to determine what aspects of the system they find useful and where they find inadequacies, or by indirect methods which measure users' performance after using the system or monitor usage of various facilities.

Interviewing users. One of the best sources of assessment information is the user population. If users feel that an explanation facility is meeting their needs, then we can say that the explanation facility is successful. Experience indicates that users are more likely to report frustrations and inadequacies with the system, but this is also useful information. As is discussed in depth in Section 4, determining what limitations of the system are responsible for the inadequacies identified by users points up areas where systems must be improved and, in expert system explanation, has inspired research efforts to alleviate the limitations.

Monitoring usage of explanation facility. Another telling assessment of an any automated tool is whether or not users actually avail themselves of the tool and whether or not their usage is successful, i.e., they are able to get the information they seek or are able to make the system perform the task they desire. To my knowledge, a study of this kind has not been done specifically in the area of explanation systems, but such studies have been done in the area of help systems. For example, an empirical study of usage of the Symbolics DOCUMENT EXAMINER, an on-line documentation system that supports keyword searches, indicated that a substantial number of interactions with the system ended in failure, especially when users were inexperienced [Young, 1987].

Impact on user's performance on task. Another way to assess the contribution of an explanation component is to determine how the explanation capabilities of the system contribute to users' effectiveness in using the system to achieve their goals. For example, if the system is intended to instruct users about how to perform some task, then it should be possible to design a simple experiment that assesses the explanation component. One way to set up such an experiment would be to have two groups of users. One group would use a version of the system with full explanation capabilities. The other group would be given the system without explanation capabilities. Both groups' performance on the task would be measured before and after using the system. If the explanation capabilities of the system are effective, we would expect the group which used the system with explanation capabilities to show greater improvement in task performance, greater retention of skills, or greater transfer of knowledge to related tasks.

3.2 Comparison to human-human advisory interactions

Fischer claims, that "human assistance, if available on a personal level, is in almost all cases the most useful source of advice and help" [Fischer, 1987]. One reason he cites for this is that most information and advice-giving systems require that users know what they are looking for when they approach such a system. However, studies of naturally occurring advisory interactions indicate that advice seekers often require assistance in formulating a query [Pollack *et al.*, 1982, Finin *et al.*, 1986].

Thus, another way to assess explanation facilities is to compare them to the ideal "explanation system", i.e., a human explainer, and to determine what capabilities of human-human explanatory interactions are present/absent from the system being evaluated. In my own work on expert system explanation, I have performed such a comparison which I discuss in detail in Section 5.

3.3 Evaluate system against set of criteria

A third approach for evaluating explanation systems would involve devising a set of evaluation criteria and rating systems accordingly. This requires not only devising the

IF: 1) the infection which requires therapy is meningitis,
2) only circumstantial evidence is available for this case,
3) the type of meningitis is bacterial,
4) the age of the patient is greater than 17 years old, and
5) the patient is an alcoholic,

THEN: there is evidence that the organisms which might be causing the infection are diplococcus-pneumoniae (.3) or e.coli (.2)

Figure 1: A MYCIN Rule With Implicit Knowledge

set of evaluation metrics, but also devising techniques for measuring the system along each dimension in an objective fashion. In Section 6, I discuss one set of evaluation criteria that has recently been proposed. In that section, we will see that devising routines to measure how a system fares against the criteria may not be an easy task, and for certain of the criteria, the measurement may inevitably remain subjective.

In the next two sections, I describe case studies using the first two methods. A set of evaluation criteria proposed for use with the third method is then discussed.

4 Method 1 - Evaluating MYCIN's Explanation Facility

One of the first studies that attempted to evaluate an explanation facility was done in the context of MYCIN, a rule-based medical consultation system designed to provide advice regarding diagnosis and therapy for infectious diseases [Shortliffe, 1976]. The rules encoding MYCIN's medical knowledge are composed from a small set of primitive functions that make up the rule language. Associated with each of the primitive functions is a template to be used when generating explanations. The English translation of a sample MYCIN rule is shown in Figure 1.

In MYCIN, a consultation is run by backward chaining through applicable rules, asking questions of the user when necessary. As a consultation progresses, MYCIN builds a *history tree* reflecting the goal/subgoal structure of the executing program. Explanations are produced from the history tree using the templates to translate the sequence of rules that were applied to reach a conclusion.

To study MYCIN's explanation facility, several scenarios in which MYCIN produced inadequate responses to questions asked by users were analyzed in order to determine the reasons for the suboptimal performance [Buchanan and Shortliffe, 1984]. As a result of this study, the implementors of MYCIN discovered several problems with the explanation facility and were able to identify limitations in the system's architecture that were responsible for these problems.

The problems identified by the implementors were:

- MYCIN could not answer some types of questions that users of the system wished to ask. Most notably, MYCIN could not produce justifications of the rules it used in making a diagnosis, i.e., MYCIN could not answer questions of the form *"Why does the conclusion of a rule follow from its premises?"*
- MYCIN could not deal with the context in which a question was asked - MYCIN had no sense of dialogue, so each question required full specification of the points of interest without reference to earlier exchanges.
- MYCIN often misinterpreted the user's intent in asking a question. The study identified examples of simple questions with four or five possible meanings depending on what the user knows, the information currently available about the patient under consideration, or the content of the earlier discussions.

The first problem reflects the lack of sufficient knowledge to support explanation, while the remaining two stem from the limitations of ad-hoc explanation techniques that fail to handle the linguistic complexities of explanation generation.

4.1 Impoverished Knowledge Bases

In attempting to adapt MYCIN for use as a tutoring system, Clancey examined MYCIN's rule base and found that individual rules served different purposes, had different justifications, and were constructed using different rationales for the ordering of clauses in their premises [Clancey, 1983b]. However, these purposes, justifications and rationales were not explicitly included in the rules; therefore many types of explanations simply were not possible and thus user questions that required such explanations as responses, could not be addressed. In particular, there were three important types of explanation that MYCIN, and other early systems that generated explanations by translating their code, could not produce.

Justifications. Early systems could not give justifications for their actions. These systems could produce only procedural descriptions of *what* they did. They could not tell the user *why* they did it or why they did things *in the order that they did them.*

For example, consider the rule shown in Figure 1 and suppose that the user wishes to know why the five clauses in its premise suggest that the organism causing the infection may be *diplococcus* or *E. coli.* MYCIN cannot explain this because the system knows no more about the association between the premises and conclusion than what is stated in this rule. In particular MYCIN does *not* know that clauses 1, 3, and 5 together embody the causal knowledge that if an alcoholic has bacterial meningitis, it is likely to be caused by *diplococcus.* Furthermore, MYCIN does *not* understand that clause 4 is a *screening clause* that prevents the system from asking whether the patient is an alcoholic when the patient is not an adult - thus making it appear that MYCIN understands this social "fact." However, MYCIN does not explicitly represent, and therefore cannot explain, this relationship between clauses 4 and 5. Even worse, not knowing that the system makes this assumption may lead the user to infer that age has something to do with the type of organism causing the infection.

Furthermore, the order in which rules are tried to satisfy a particular goal will affect the order in which subgoals are pursued. When a goal such as determining the identity of the organism is being pursued, MYCIN invokes all of the rules concerning the identity of the organism in the order in which they appear in the rule base. This order is determined by the order in which the rules were entered into the system! Thus MYCIN cannot explain why it considers one hypothesis before another in pursuing a goal. In addition, because attempting to satisfy the premises of a rule frequently causes questions to be asked of the user, MYCIN cannot explain why it asks questions in the order it does because this too depends on the ordering of premises in a rule and the ordering of rules in the knowledge base. As Clancey points out [Clancey, 1983b], "focusing on a hypothesis and choosing a question to confirm a hypothesis are not arbitrary in human reasoning" and thus users will expect the system to be able to explain why it pursues one hypothesis before another and will expect questions to follow some explicable line of reasoning.

Explications of General Problem Solving Strategy. The knowledge needed to explain general problem-solving strategies was not explicitly represented in early systems. In MYCIN, because several different types of knowledge are confounded in the uniformity of the rule representation, it is difficult to identify and explain MYCIN's overall diagnostic strategy.

Again, consider the rule shown in Figure 1. Another hidden relationship exists between clauses 1 and 3. Clearly bacterial meningitis is a type of meningitis, so why include clause 1? The ordering of clauses 1 and 3 implicitly encodes strategic knowledge of the consultation process. The justification for the order in which goals are pursued is implicit in the ordering of the premises in a rule. The choice of ordering for the premises is left to the discretion of the rule author and there is no mechanism for recording the rationale for these choices. This makes it impossible for MYCIN to explain its general problem-solving strategy, i.e., that it establishes that the infection is meningitis before it determines if it is bacterial meningitis because it is following a refinement strategy of diagnosing the disease [Hasling *et al.*, 1984].

Definitions of Terminology. Terminological knowledge was also not explicitly represented in MYCIN or other early systems. Users who are novices in the task domain will need to ask questions about terminology to understand the system's responses and to be able to respond appropriately when answering the system's questions. Furthermore, experts may want to ask such questions to determine whether the system's use of a term is the same as their own. Because the knowledge of what a term means is implicit in the way it is used, the system is not capable of explaining the meaning of a term in a way that is acceptable to users. An effort to explain terms by examining the rule base of an expert system [Rubinoff, 1985] has been only partially successful because so many different types of knowledge are encoded into the single, uniform rule formalism. This makes it difficult to distinguish definitional knowledge from other types of knowledge.

4.1.1 Summary

As we have seen, rules and rule clauses incorporate many different types of knowledge, but the uniformity of the rule representation obscures their various functions thus making comprehensible explanation impossible. Much of the information that goes into writing the rules or program code that make up an expert system including justification, strategic information, and knowledge of terminology is either lost completely or made implicit to the extent that is no longer available to be explained. Both Swartout (1983) and Clancey (1983) have argued that the different types of knowledge (definitional, world facts, causal, strategic) must be separately and explicitly represented if systems are to be able to explain their strategy and justify their conclusions. As a consequence, later efforts have addressed the problem of capturing the knowledge and decisions that went into writing the program and explicitly representing this information so that it will be available for explanation. This research is briefly discussed in Section 7. For a more detailed discussion of knowledge representation issues and a description of the Explainable Expert Systems approach, see [Swartout and Smoliar, 1987b, Swartout and Smoliar, 1987a].

4.2 Inadequate Natural Language Techniques

The other problems identified by the MYCIN implementors in their analysis of inadequate responses stem from limitations in the natural language capabilities of MYCIN. Specifically, question understanding and interpretation procedures are limited, thus restricting the kinds of questions that may be asked and the manner in which they must be phrased. To avoid the difficult problems of inferring the user's intent in asking a question, MYCIN interprets a user's "Why?" query in only one way, even though it could have a variety of meanings, such as "Why is it important to determine....?," "Why did you ask about that instead of....?," "Why do you ask that now?," or "Why does the conclusion follow from the premises?" All of these interpretations are valid questions about the system's knowledge and behavior, yet MYCIN always assumes the first interpretation and does not allow the other questions to be asked.

For example, consider the explanation produced by MYCIN in Figure 2. When the user asks "Why?" the second time, the system assumes that the user is asking *"Why is it important to determine the category of ORGANISM-1?"*. But, the user may really be asking a very different type of question, namely *"Why is it the case that a gram negative, facultative rod acquired in a hospital setting is likely to be pseudomonas?"*[1] We have already seen that the causal knowledge needed to answer this question is not represented in MYCIN. But even if it were, MYCIN could not determine what "why?" question was being asked in a given context because MYCIN does not view the explanation session as an on-going dialogue. Each question-answer pair is viewed independently and references to previous portions of the dialogue can be made only in stilted and artificial ways. A model of explanation that addresses this problem is described in Section 5.3.

[1]This example is adapted from an example found in [Davis, 1976].

Was the infection with ORGANISM-1 acquired while the patient was hospitalized?

****WHY?**

[i.e., WHY is it important to determine whether or not the infection with ORGANISM-1 was acquired while the patient was hospitalized?]

[1.0] This will help to determine the identity of ORGANISM-1.

It has already been established that

[1.1] the gram stain of ORGANISM-1 is gramneg, and

[1.2] the morphology of ORGANISM-1 is rod, and

[1.3] the aerobicity of ORGANISM-1 is facultative

Therefore, if

[1.4] the infection with ORGANISM-1 was acquired while the patient was hospitalized

Then

there is weakly suggestive evidence (.2) that the identity of the organism is pseudomonas

[RULE050].

****WHY?**

[i.e., WHY is it important to determine the identity of ORGANISM-1?

...

Figure 2: A Sample MYCIN Explanation

4.2.1 Summary

The experience with the MYCIN study shows that asking users to indicate unsatisfactory explanation behavior is a very useful method for evaluation. This study pointed out several scenarios where MYCIN's explanation facility did not behave as users expected or could not give an explanation they desired. Analyzing these scenarios enabled MYCIN's implementors and other researchers to discover the architectural limitations that were responsible for the inadequacies and spurred research aimed at improving the problems.

5 Method 2 - Comparison to human-human interactions

My own work in expert system explanation was motivated by an observation of a great disparity between what analyses of naturally occurring advisory dialogues reveal, on the one hand, and the explanation facilities that current systems provide and the assumptions they make about how users interact with experts, on the other. Most systems make the tacit assumption that the explanations they produce will be understood by their users. However, in human-human advisory situations, ***people almost always ask follow-up questions!*** Expert and advice-giving systems are expected to provide solutions and advice to users faced with real problems in complex domains. They often produce complex multisentential responses, such as definitions of terms, justification of results, and comparisons of alternate methods for solving problems. Users must be able to ask follow-up questions if they do not understand an explanation or want further elaboration. Answers to such questions must take into account the dialogue context.

5.1 What the data reveals

In a study of a "naturally occurring" expert system, Pollack *et al.* (1982) found that user-expert dialogues are best viewed as a negotiation process in which the user and expert negotiate the statement of the problem to be solved in addition to a solution that the user can understand and accept. In my own work on explanation, I examined samples of naturally occurring dialogues from several different sources: tape-recordings of office-hour interactions between first year computer science students and teaching assistants, protocols of programmers interacting with a mock program enhancement advisor, and transcripts of electronic dialogues between system users and operators taken from [Robinson, 1984]. A portion of a dialogue extracted from the transcripts I collected appears below:

TEACHER	OK, so what is it, it's using stacks, right?
STUDENT	Yea, well, cause, um, aren't we supposed to use linked lists?
TEACHER	You don't have to use linked lists. You don't.
STUDENT	But OK. You said stacks, right?
TEACHER	In LISP we implemented stacks as linked lists. In C we can implement stacks as an array.
STUDENT	Wait, in LISP...
TEACHER	In LISP, we implemented, past tense, implemented stacks as linked lists. Right? In C, we can do it anyway we want. We can implement it as linked lists or as arrays, uh, I don't know any obvious data structures after that. But, um, you can use a linked list or an array. I would use an array, personally.

In this dialogue, the student does not understand the difference between the general concept of a stack as a way of managing data and its implementation using a particular data structure. The teacher thinks she has cleared up the student's misunderstanding with her first explanation, but in fact she has not, as indicated by the student's "Wait" and hesitation. The teacher enhances her earlier response by emphasizing the point she made in the previous explanation, and then elaborating on the notion that a stack can be implemented using various data structures. Similarly, automated systems must be able to offer further elaborations of their responses or alternate explanations, even when the user is not very explicit about which aspect of the explanation was not clear.

An analysis of dialogues such as this one led to the following observations:

- *Users frequently do not fully understand the expert's response.* Users rarely stated a problem, received a result or explanation, and then left, satisfied that they understood and accepted the expert's explanation. The expert frequently found the need to define terms or establish background information in response to feedback that the listener did not completely understand the response.

- *Users frequently ask follow-up questions.* Users frequently requested clarification, elaboration, or re-explanation of the expert's response.

- *Users often don't know what they don't understand.* Users frequently could not formulate a clear follow-up question. In many cases, the follow-up question was vaguely articulated in the form of mumbling, hesitation, repeating the last few words of the expert's response, or simply stating "I don't understand." Often the expert did not have much to go on, but still was required to provide further information.

- *Experts do not have a detailed model of the user.* From the dialogues I examined, it is clear that experts do not have a complete and correct model of their users. While we can safely assume that experts have some model of the users, it seems that since many and varied users are likely to seek their help, this model is more likely to be a stereotypic model that may be incomplete or incorrect for any given hearer than a detailed model of any individual. Yet, as the dialogues show, the user and the expert are able to communicate effectively.

5.2 The State of Conventional Explanation Systems

Studies such as these and many others [Finin *et al.*, 1986, Suchman, 1987, Cawsey, 1989] show that explanation requires dialogue, yet few intelligent systems participate in a dialogue with their users. The explanation facilities of most current systems can be characterized as:

- **unnatural**: explanation generation does not employ linguistic knowledge about how texts should be constructed. The unnatural structure of the resulting texts often obscures the important points in an explanation.

- **inextensible**: new explanation strategies cannot be added easily.

- **unresponsive**: the system cannot answer follow-up questions or offer an alternative explanation if a user does not understand a given explanation.

- **insensitive**: explanations do not take the context into account. Each question and answer pair is treated independently.

- **inflexible**: explanations can be presented in only one way.

Generally, these problems stem from the fact that explanation generation has not been considered as a problem requiring its own expertise and worthy of an architecture that supports a sophisticated problem-solving activity. More specifically, these problems stem from limitations in several areas as outlined below.

Many expert systems produce explanations from a trace of the expert system's line of reasoning. Much effort has gone into identifying clever strategies for annotating, pruning, traversing, and translating the execution trace to produce 'good' explanations (e.g., [Weiner, 1980]). However, there are several problems with this approach. First, it places much of the burden of producing explanations on expert system builders and their ability to structure the rules or program code in a way that will be understandable to users who are knowledgeable about the task domain. For example, in MYCIN, an attempt was made to make each rule an independent "chunk" of medical knowledge reflecting a complete, coherent explanation. In addition, as other researchers ([Davis, 1976, Webber and Joshi, 1982, Swartout, 1983, Clancey, 1983b, Pavlin and Corkill, 1984]) have noted, the computationally efficient reasoning strategies used by expert systems to produce a result often *do not* form a good basis for understandable explanations. Experience has shown that explanations produced by paraphrasing the system's execution trace correspond too literally to the program structure, which is dictated, at least in part, by implementation considerations which may obscure the underlying domain-related reasoning. Moreover, there is no reason to assume that a simple paraphrase of the program's execution will produce an explanation that conforms to the linguistic conventions dictating discourse structure and coherence.

Consider the explanation in Figure 3 which was produced by NEOMYCIN. As will be discussed further in Section 7.1, NEOMYCIN [Clancey and Letsinger, 1981] was developed as part of an effort to build an intelligent tutoring system to instruct medical students in diagnosis. For this purpose, it was necessary for the system to be able explain its overall diagnostic strategy. Recall that MYCIN could not answer questions of this type because of limitations in its knowledge base. In NEOMYCIN domain-independent problem-solving knowledge is represented explicitly and separately from knowledge about the domain. Because its explanations are based on a more explicit representation, NEOMYCIN is able to produce explanations of its problem-solving strategies that are a significant improvement over those of MYCIN. For example, NEOMYCIN is able to give abstract explanations of its general problem-solving strategy, see [Hasling *et al.*, 1984] for details. However, the explanation in Figure 3 demonstrates that, even with the improved knowledge base of NEOMYCIN, simply paraphrasing the system's reasoning often produces text that is unsuitable for users. The problem with this explanation is that its structure corresponds too literally to the structure of the method for achieving the task, i.e., applying one rule after another. While the explanation repeats that it "gave thorough consideration to any recent findings" *four times*, the explanation does not make clear the overall strategy of applying rules that strongly contribute to the current hypothesis (e.g., causal precursors) before applying rules that are weaker indicators (e.g., general evidence). What is needed is a more sophisticated explanation strategy that could recognize the similarity in the four rule applications and structure

**HOW 19.0?

[i.e. How are we trying to decide whether Mary has tuberculosis?

We did the following:

[20.1] Applied a rule mentioning a causal precursor to tuberculosis (tb risk factor; rule525), and

[20.2] Gave thorough consideration to any recent findings.

[20.3] Applied a rule mentioning general evidence for tuberculosis (focal neurological signs; rule366), and

[20.4] Gave thorough consideration to any recent findings.

[20.5] Applied a rule mentioning general evidence for tuberculosis (compromised host; rule309), and

[20.6] Gave thorough consideration to any recent findings.

We are now:

[20.7] Applying a rule mentioning general evidence for tuberculosis (a history of granulomas on biopsy of the liver, lymph node, or other organ; rule002).

We will next:

[20.8] Give thorough consideration to any recent findings.

[back to question 14 ....]

Figure 3: A Sample NEOMYCIN Explanation

this information accordingly. This example is symptomatic of the more general problem that simply paraphrasing the system's knowledge base or reasoning trace is not guaranteed to produce understandable explanations, *regardless of how well that knowledge base is structured.* In order to produce natural language explanations that are coherent to human users, an explanation facility must have linguistic knowledge about discourse structure and strategies for employing that knowledge to achieve its explanatory goals.

Another problem is that the explanation components of conventional expert systems are difficult to extend. Because explanation is typically implemented in ad-hoc procedures or large collections of rules, it is difficult to understand how adding new rules or procedures will interact with existing facilities. In general, answering a new type of question involves coding new procedures or building new templates from scratch. Little, if any, of the existing code is useful. Moreover, expert systems are becoming more complex as system builders augment their knowledge bases to include the underlying support knowledge needed to allow systems to answer a broader range of questions about their domain knowledge and behavior. The knowledge bases of newer expert systems separate different types of knowledge [Clancey and Letsinger, 1981, Neches *et al.*, 1985, Swartout and Smoliar, 1987b] and represent knowledge at various levels of abstraction [Patil, 1981], and thus confront explanation generators with an array of choices about what information to include in an explanation and how to present that information which were unavailable in the simple knowledge bases of earlier systems. Meeting the challenge posed by these richer knowledge bases will require more sophisticated and linguistically motivated explanation generators.

Researchers in computational linguistics have addressed the issues of selecting information from a complex knowledge base and organizing it into a text that adheres to the conventions of discourse structure (cf. [Weiner, 1980, McKeown, 1982, Appelt, 1981, McCoy, 1985, Reichman, 1981]. Results from this research have been successfully incorporated into recent explanation systems [Cawsey, 1989, McKeown, 1988, Moore and Swartout, 1989, Paris, 1990, Wolz, 1990] to enable these systems to produce natural explanations.

However, few systems are capable of responding to follow-up questions. The problem is that most intelligent systems that respond to user's questions view generating responses as a *one-shot process*. That is, they assume that they will be able to produce an explanation that the user will find satisfactory in a single response. This one-shot approach is inconsistent with analyses of naturally occurring advisory dialogues. Moreover, if a system has only one opportunity to produce a text that achieves the speaker's goals without over- or under-informing, boring or confusing the listener, then that system must have an enormous amount of detailed knowledge about the listener. Taking the one-shot approach has led to a view that improvements in explanation will follow from improvements in the *user model*. Recognizing this need, researchers have studied how to build user models (e.g., [Kass and Finin, 1988, Rich, 1989, Chin, 1989, Kobsa, 1990, Kobsa, 1989]) and how to exploit them to produce the 'best' possible answer in one shot. Considerable effort has been expended on building complex user models, containing large amounts of detailed information about a user, including the user's goals and plans, attitudes, capabilities, preferences, level of expertise, beliefs, beliefs about the system's beliefs (and other such mutual beliefs). From these models, a system then attempts to generate the 'best possible' answer for that particular user. Already, systems employing such models have demonstrated that a user model can be used to guide a generation system in producing answers that appear to be appropriately tailored to the user (e.g., [Appelt, 1985, Kass and Finin, 1988, McCoy, 1985, Paris, 1988, van Beek, 1986, Wolz, 1990]).

While the quality of explanations can be demonstrably improved by employing a user model, a system that is critically dependent on such a model will not suffice. Sparck Jones (1984) questions whether it is even feasible to build complete and correct user models. To date, no robust system for automatically acquiring these complex and detailed user models exists, and hand-crafting them is time-consuming and error-prone. The completeness and accuracy of a user model cannot be guaranteed. Thus, unless mechanisms are developed by which systems can dynamically acquire and update a user model by interacting with the user, the impracticality of building user models will prevent much of the work on tailoring from being successfully applied in real systems. Some researchers have begun to develop tools to aid systems in acquiring user models. For example, Kass' General User Modeling Acquisition Component (GUMAC) [Kass, 1988] provides a set of domain-independent acquisition rules that allows a system to acquire a model of the user by drawing inferences from the user's utterances during a dialogue with an expert system. Although such tools show promise, systems that *rely* on correct and complete user models are likely to be brittle. Indeed, most of Kass' acquisition rules require that the user and system be able to carry on an initial dialogue (for example, to gather data or establish the problem to be solved) during which the user model is being acquired. Thus the system must be able to carry on a dialogue with the user *in order to acquire the user model.* Clearly the system must be able to communicate without a complete and correct model if Kass' system is to be feasible.

More importantly, by focusing on user models, researchers have ignored the rich source of guidance that people use in producing explanations, namely feedback from the listener [Ringle and Bruce, 1981].

5.3 What is needed in a dialogue system

In my own work, I have developed a model of explanation in which feedback from the user is an integral part of the explanation process. In designing this model, I identified the capabilities that a system must possess in order to participate in a dialogue. They are as follows.

Accept feedback from the user. To be responsive to a user's feedback, a system must allow the user to provide that feedback. Many systems do not have a means for allowing the user to indicate dissatisfaction with a given explanation. The user who cannot ask one of a prescribed set of follow-up questions in a prescribed form, is without recourse.

The system must understand its own explanations. In order to provide elaborating explanations, clarify misunderstood explanations and respond to follow-up questions in context, a system must view the explanations it produces as objects to be reasoned about later. In particular, detailed knowledge about how the explanation was "designed" must be recorded, including: the goal structure of the explanation, the roles individual clauses in the text play in responding to the user's query, how the clauses relate to one another rhetorically, and what assumptions about the listener's knowledge may have been made. Without this knowledge, a system cannot be sensitive to dialogue context or responsive to the user's needs.

Interpret questions taking previous explanations into account. When participating in a dialogue, a system must realize that the same question can mean different things in different contexts, i.e., the system must be able to interpret questions taking into account what the user knows, the information available about the current problem-solving situation, or the content of the previous discussion.

Have multiple strategies for answering a question of a given type. Finally, making oneself understood often requires the ability to present the same information in multiple ways or to provide different information to illustrate the same point. Current systems are inflexible because they typically have only a single response strategy associated with each question type, instead of the sophisticated repertoire of discourse strategies that human explainers utilize. Without multiple strategies for responding to a question, a system cannot offer an alternative response even if it understands why a previous explanation was not satisfactory.

Based on these requirements, I devised a model for explanation intended to alleviate the limitations described above. The model captures the explainer's reasoning about the design of an explanation and utilizes this knowledge to effectively support dialogue. The main features of this model are:

- explanation knowledge is represented explicitly and separately from domain knowledge in a set of strategies that can be used to achieve the system's discourse goals
- the system has many and varied strategies for achieving a given discourse goal
- utterances are planned in such a way that their intentional and rhetorical structure is explicit and can be reasoned about
- the system keeps track of conversational context by remembering not only what the user asks, but also the planning process that led to an explanation
- information in the user model is utilized when it is available, but the system is able to operate effectively even when no pertinent information appears in the user model or when the user model is incorrect

This model has been implemented in an explanation generation facility for the Explainable Expert Systems (EES) framework [Neches *et al.*, 1985, Swartout and Smoliar, 1987b], a domain-independent shell for creating expert system applications. When an expert system is built in EES, an extensive development history is created that records the domain goal structure and design decisions behind the expert system. This structure, as well as the system's static knowledge base, and the execution trace produced when the system is used to solve a particular problem, are all available for use by the explanation facility.

A detailed description of the implementation of the explanation facility is beyond the scope of this paper, but may be found in [Moore and Swartout, 1989, Moore, 1989]. Briefly, the explainer works in the following way. When the system needs to communicate with the user, a discourse goal (e.g., make the hearer know a certain concept, persuade the hearer to perform an action) representing an abstract specification of the response to be produced is posted. The system then uses its explanation strategies to plan an explanation to achieve the goal. When planning is complete, the system presents the explanation to the user, recording the *text plan* that produced it in a dialogue history. The text plan captures the "design" of an explanation, indicating what the system was trying to explain, how it explained it, and what alternative ways could have been used to explain the same thing. The dialogue history is a record of the conversation that has occurred thus far and includes the user's utterances as well as the text plans that lead to the system's responses. The text plans recorded in the dialogue history are used to

SYSTEM	What characteristics of the program would you like to enhance?	[1]
USER	Maintainability.	[2]
	⋮	
SYSTEM	You should replace `(SETQ X 1)` with `(SETF X 1)`.	[3]
USER	Why?	[4]
SYSTEM	I'm trying to enhance the maintainability of the program by applying transformations that enhance maintainability. **SETQ**-to-**SETF** is a transformation that enhances maintainability.	[5]
USER	Why?	[6]
SYSTEM	Transformations that enhance maintainability are defined as transformations whose right-hand-side's use is more general than its left-hand-side's use. **SETF** can be used to assign a value to any generalized-variable. **SETQ** can only be used to assign a value to a simple-variable.	[7]
USER	What is a generalized-variable?	[8]
SYSTEM	Simple-variables are storage locations named by symbols. This concept can be generalized to a storage location that can be named by any access function. These storage locations are called generalized-variables.	[9]
USER	Huh?	[10]
SYSTEM	For example, the *car* and *cdr* of a cons are generalized-variables, named by the access functions **CAR** and **CDR**. Other examples are an element of an array or a component of a structure.	[11]
USER	OK.	[12]

Figure 4: Sample Dialogue

provide the conversational context necessary to handle follow-up questions and recover when feedback from the user indicates that the system's explanation is not satisfactory.

We have found that the text plans that produced previous explantions are indispensable in determining how to interpret the hearer's feedback. Figure 4 shows a sample dialogue with the Program Enhancement Advisor (PEA), a prototype expert system implemented within the EES framework in order to test our explanation facility. PEA is an advice-giving system intended to aid users in improving their Common LISP programs by recommending transformations that enhance the user's code. This dialogue demonstrates that our system is able to participate in an on-going dialogue with the user.

First, the two why-questions appearing on lines [4] and [6] are interpreted differently because they appear in two different contexts. The first why-question occurs after the system has recommended that the hearer perform an action (line [3]). The system interprets this question as a request from the user to be persuaded that he should perform this action. One of the strategies the system has for persuading hearers to perform actions is to state the shared goal (in this case `enhance maintainability`) that led to its recommending the action, to state the method being applied to achieve this goal (`apply transformations that enhance maintainability`), and finally to state how the recommended act is involved in achieving the goal. This strategy leads to the system's response on line [5]. When the user asks "Why?" again after this response (line [6]), it is interpreted as a request for the system to justify the last statement it made in its explanation, leading to the interpretation "Why is SETQ-to-SETF is a transformation that enhances maintainability?"

Another important aspect of this dialogue is that our system allows the user to ask the vaguely articulated question "Huh?" on line [10]. In order to answer the user's why-question on line [6], the system must explain why SETQ-to-SETF is a transformation that enhances maintainability. In doing so, the system uses the term `generalized variable`, which is apparently unfamiliar to the user as evidenced by the follow-up question "What is a generalized variable?" on line [8]. In answering this question, the system uses one of its many strategies for describing a concept. In particular, it uses a strategy which reminds the user of a familiar concept (`simple variable`) and which is a specialization of the concept being explained (`generalized variable`). The system then abstracts from the known concept to the new concept, focusing on the aspects (in this case, `use`) of the more specific concept that are being generalized to form the more general concept which it then names. This is a very general strategy for describing a concept that can be applied whenever the user is familiar with a specialization of the concept to be described. In this case, the user does not understand this explanation, but cannot ask a pointed question elucidating exactly what is not understood. In our system, one of the options available to the user is to simply ask "Huh?", indicating that the previous explanation was not understood. The system "recovers" from this type of "failure" by finding another strategy for achieving the failed goal. In this case, the failed goal is to describe a concept, and the system recovers by giving examples of this concept, line [11].

A more detailed discussion of how this dialogue is produced may be found in [Moore, 1989]. We have also demonstrated that the text plans recorded in the dialogue history can be used to select the perspective from which to describe or compare objects [Moore, 1989], as well as to avoid repeating information that has already been communicated [Moore and Paris, 1989]. In [Moore and Swartout, 1990], we show how the information in text plans allows the system to provide an intelligent hypertext-style interface in which users highlight the portion of the system's explanation they would like clarified, and the system produces a context-sensitive menu of follow-up questions that may be asked at the current point in the dialogue.

5.4 Discussion

As was the case with Method 1, comparing the behavior of an explanation system to that of human explainers has proven useful. Characterizing the differences and understanding the reasons for the disparities allows us to identify specific aspects of explanation systems that must be improved.

One issue that must be addressed when using this method for assessing explanation systems is how far to carry the analogy between human-human interactions and human-machine interactions. Certainly there are differences between the capabilities of humans and machines and we should not blithely assume that human-computer interaction must mimic human-human interaction in all aspects. Rather, we must make reasoned decisions about what aspects of human-human interaction we wish to preserve in human-machine interaction, what differences can be tolerated, and in what ways human-computer interaction can improve upon human-human interaction.

In some cases, intelligent systems may offer benefits beyond the scope of human explainers. For example, one way in which expert systems differ from human experts is that, if built according to certain principles, expert systems have access to their problem-solving strategies and can accurately report exactly what methods were used to solve a problem and why these methods were chosen (see for example [Neches *et al.*, 1985, Swartout and Smoliar, 1987b]). Human experts on the other hand, do not have access to the actual methods which they employ in reasoning. They can, however, construct a justification for why a solution is correct or reconstruct a plausible chain of reasoning based on their rich model of the domain. Within the expert system explanation community, there is currently a debate about how closely the "line of explanation" must follow the system's "line of reasoning". Wick and Thompson (1989) argue that an effective explanation need not be based on the actual reasoning processes that the system used in solving the problem, but rather, the system's results may be supported by other sources of information about the domain. However, as we will see in Section 6, Swartout argues that such an approach is at odds with one of the desired attributes for expert system explanation, namely *fidelity*, which requires that the explanation be an accurate representation of what the expert system really does. Regardless of how this issue is resolved, the fact that expert systems may have access to their line of reasoning affords an opportunity not available to human experts.

The hypertext-style pointing interface we designed for our explanation facility provides another example of the way in which human-computer interaction may usefully differ from human-human interaction. We designed this interface to provide users with a convenient way to ask questions about previously given explanations. In human-human interactions, people can ask questions that refer to previous explanations with utterances such as *"I didn't understand that part about applying transformations that enhance maintainability?"* But such questions pose a difficult challenge for natural language understanding because such questions often intermix meta-level references to the discourse with object-level references to the domain. Our hypertext-like interface allows users to point to the portion of the system's explanation that they would like clarified. By allowing users to point, many of the difficult referential problems in natural language analysis can be avoided. Implementing this interface was possible in our explanation facility precisely because our system understands its own explanations and thus is able to understand what the user is pointing at. The system then offers the user a menu of follow-up questions that it knows how to answer and that make sense in the current context. While this interface differs from what occurs in human-human interaction, it provides a pragmatic solution to the problem of allowing users to ask questions about previous explanations. It may even be the case that when interacting with a computer, users prefer to highlight the text they would like to ask about and to receive a menu of possible questions, rather than attempting to formulate a natural language question. We plan to test this hypothesis in future work.

6 Method 3 – Criteria for Evaluating Explanation Systems

The third method for assessing explanation systems is to devise a set of criteria for evaluation and to rate systems according to these criteria. Recently, Swartout has attempted to codify a set of desiderata for explanation facilities [Swartout, 1990]. He has suggested that these requirements can be used as metrics for evaluating the performance of explanation systems and progress in the field. The desiderata, shown in Figure 5, fall into three classes. The first places constraints on the mechanism by which explanations are produced. The second and third specify requirements on the explanations themselves. The fourth and fifth are concerned with the effects of an explanation facility on the construction and execution of the expert system of which it is a component.

Swartout has gone on to describe the implications of some of these criteria. For example, the need for fidelity has several implications. First, the explanations must be based on the same underlying knowledge that the system uses for problem solving. Thus systems that produce explanations using canned text or 'fill-in-the-blank' templates would receive a poor rating because there can be no guarantee that explanations produced by these systems are consistent with the program's behavior. Another implication of the fidelity criterion is that the expert system's inference engine should be as simple as possible with a minimal number of special features built into the interpreter. Such features are not part of the explicit knowledge base of the system, and are either not explained at all or are explained by special-purpose routines built into the explanation facility. This introduces the potential for inaccuracy because changes made to the interpreter require that changes be made independently to the explanation routines. For example, the certainty factor mechanism that manages MYCIN's reasoning about uncertainty cannot be explained because it is built into MYCIN's interpreter.

The efficiency metric has important implications as well. It would rate a system that provided good justifications of its actions as poor if that system did so by always reasoning from first principles. Such a system would be re-deriving its expertise on each run and would be very inefficient.

6.1 Discussion

The criteria proposed by Swartout are very comprehensive and are quite useful as qualitative guidelines. However, it would be desirable to form evaluation metrics from these criteria with objective methods for assigning ratings to an explanation system. In some

1. **Fidelity.** Explanations must be an accurate representation of what the expert system really does.

2. **Understandability.** Explanations must be understood by users. Understandability is not a single factor, but is made up of several factors, including:

   - **Terminology.** The terms used in explanations must be familiar to the user or the system must have the capability to define them.
   - **Abstraction.** The system must be able to give explanations at different levels of abstraction of terminology. For example, in describing a patient's problem, the system should be able to use the abstract term bacterial infection, or the more specific term *E. coli* infection.
   - **Summarization.** The system must be able to provide descriptions at different levels of detail.
   - **Viewpoints.** The system must be able to present explanations from different points of view that take into account the user's interests and goals
   - **Linguistic Competence.** The system should produce explanations which sound "natural".
   - **Coherence.** Taken as a whole, the explanations should form a coherent set. Explanations should take into account previous explanations.
   - **Composability.** When several things must be conveyed in a single explanation, there should be smooth transitions between topics.
   - **Correct Misunderstandings.** The system must allow the user to indicate that an explanation is unsatisfactory and be capable of providing further clarification.

3. **Sufficiency.** Enough knowledge must be represented in the system to support production of the kinds of explanations that are needed. The system must be able to handle the types of questions that users wish to ask.

4. **Low Construction Overhead.** Providing explanation should impose light load on expert system construction, or any load that is imposed should be recovered by easing some aspect of the expert system lifecycle (e.g., maintenance or evolution).

5. **Efficiency.** The explanation facility should not degrade the runtime efficiency of the expert system.

Figure 5: Swartout's Desiderata for Explanation Systems

cases, the task of devising a method for assigning a value to the metric seems straightforward. For example, fidelity can be assessed by comparing traces of the system's problem-solving with the natural language explanations it produces. Techniques from software engineering could be helpful in estimating the overhead in system construction due to the explanation facility and how much savings in the maintenance and evolution cycles are due to design decisions attributable to the requirements imposed by explanation. Cases in which systems have been re-engineered to provide better explanation capabilities also offer opportunities for evaluation. For example, since NEOMYCIN was developed to address certain of the limitations in MYCIN's explanation capabilities, we can get an estimate of runtime efficiency by comparing NEOMYCIN to MYCIN solving the same diagnostic problem. Moreover, the experiences of knowledge engineers working on both systems should give insight into whether the restructuring of the knowledge base for NEOMYCIN aids in maintenance or evolution.

Some aspects of understandability could also be objectively measured. For example, one way to evaluate the factor of composability (smoothness between topic transitions in a single explanation) would be to analyze the system's explanations to determine whether they adhere to constraints governing how focus of attention shifts, as defined by Sidner (1979) and extended by McKeown (1982).

In other cases, it is difficult to envisage how objective measures for assessment could be devised. For example, how can we assign a value to an explanation's naturalness (linguistic competence) and coherence? The ratings of such factors are inevitably subjective and can only be judged by human users. Furthermore, what is understandable to one user may be obscure to others. The most promising way to assess the understandability of a system's explanation will involve techniques such as those included in the discussion of Method 1, i.e., assessing the users' satisfaction with the explanations or the impact of the explanations on users' performance.

The criteria proposed by Swartout provide a good starting point for devising metrics for assessment, but clearly much more work needs to be done. In the next section, I discuss two examples of improved architectures for explanation.

7 Explainable Architectures for Expert Systems

The insights gained from analyses of the inadequacies of early expert systems led researchers to attempt to design architectures for expert systems that would improve their explanation capabilities. In designing new architectures, researchers had the goals of capturing the knowledge needed to support the types of explanation users desire, and to structure that knowledge appropriately. Here I briefly discuss two such systems.

7.1 NEOMYCIN

NEOMYCIN [Clancey and Letsinger, 1981] was developed in order to teach medical students about diagnosis, and for this purpose it was necessary to be able to justify the diagnostic associations encoded in MYCIN's rules and to explicate the overall diagnostic strategy of gathering information and focusing on hypotheses. Recall that MYCIN could not answer questions of these types due to limitations in its knowledge base. NEOMYCIN was designed with the goal of capturing control knowledge more explicitly so that it could be explained and re-used. Clancey has argued that NEOMYCIN's metarules constitute a domain-independent diagnostic strategy that could be applied to related problems in other domains [Clancey, 1983a].

In NEOMYCIN, a domain-independent diagnostic strategy is represented explicitly and separately from knowledge about the domain (the disease taxonomy, causal and data/hypothesis rules, and world facts). To build an expert system using NEOMYCIN, the developer must first identify the "task" structure of the problem, e.g., *make-diagnosis*, *pursue-hypothesis*, *explore-and-refine*. A diagnostic strategy is then represented as a set of *tasks*, which are meta-level goals, and *meta-rules* [Davis, 1980] for achieving these goals. An ordered collection of metarules defines a *generic procedure* for achieving a task. Next, domain-specific rules are organized into rule-sets based on this task structure. Rule-sets become active depending on which tasks have been posted.

Because NEOMYCIN's strategic knowledge is explicitly represented, the system can produce explanations of its problem-solving strategies. For example, Figure 6 (from [Hasling *et al.*, 1984]), shows that NEOMYCIN is able to give abstract explanations of its general problem-solving strategy. In addition, the possibility of knowledge re-use exists if indeed NEOMYCIN's metarules constitute a domain-independent diagnostic strategy.

However, above we saw an example of NEOMYCIN's explanation behavior that indicated that the improved representation of strategic knowledge alone was not sufficient to improve all types of explanations (see Figure 3). This example showed that more work must be done to improve the understandability of the system's explanations. In particular, techniques for summarization and choosing the appropriate level of abstraction for an explanation in a particular situation must be devised. The efficiency of the system is also an issue that must be evaluated, since NEOMYCIN essentially "re-discovers" its expertise each time it solves a problem.

7.2 Explainable Expert Systems (EES) Framework

EES [Neches *et al.*, 1985, Swartout and Smoliar, 1987b], like its precursor XPLAIN [Swartout, 1983], grew out of the observation that much of the knowledge needed to produce explanations is not explicitly represented in the expert system's knowledge base, but instead remains in the system builder's head. The goal of EES was to create framework, or 'shell', that captures this knowledge. The approach taken in designing the EES framework was to first identify the types of explanations that must be produced. This, in turn, indicated the kinds of knowledge that must be represented. The EES language allows a system developer to provide knowledge about an application domain, mainly in terms of an abstract specification of how the domain works, how problems are solved in the domain, and domain terminology. The system builder is then replaced by an automatic programmer that synthesizes the desired expert system from this specification, recording all the design decisions made during the program-writing process. The explanation routines then use the recorded design history to provide richer explanations, including

17) Has Mary been hospitalized recently?

****WHY?**

⋮

****HOW 25.0?**

[i.e. HOW are we trying to develop the differential using basic history and pertinent physical exam?

We did the following:

[26.1] Got a general idea of the problem: categorized it into one of several pathogenic classes or disease loci, or both.

[26.2] Confirmed and refined the differential diagnosis through specific questions.

We are now:

[26.3] Rounding out the diagnostic information by looking generally into past medical history and by reviewing systems.

[back to question 17 ....]

****NO**

Figure 6: A Sample NEOMYCIN Explanation

justications for actions.

A sample of the explanations produced by the PEA system which was built within the EES framework was shown in Figure 4. These explanation capabilities were possible for two reasons. First, the EES framework provides the types of knowledge needed to support explanation. Second, explanation is treated as a sophisticated problem-solving activity requiring its own knowledge and expertise. Techniques from natural lanaguage generation and new techniques for dialogue management were incorporated into the explanation facility [Moore, 1989]. Other expert systems are being developed using EES [Paris, 1990] and research on the explanation facility is continuing.

In addition to its implications for explanation, we have found that the EES approach offers other advantages related to development and maintenance. For example, the discipline imposed by the knowledge representation in EES, provides guidance for knowledge engineers during the development process. We believe that this rigor will make errors and inconsistencies in the knowledge base easier to detect. In addition, because the automatic program writer creates an executable expert system, it is the compiled code that is actually executed, but the rationale for that code is available for explanation. Thus representing the knowledge needed for explanation does not incur runtime overhead; the system is *not* re-deriving its expertise on every run. In terms of construction overhead, it is clearly more work to develop an expert system using EES since more knowledge must be represented than in a system such as MYCIN. However, we have found that maintenance and evolution are facilitated since modifications are performed at the knowledge base (i.e., "specification") level, rather than at the implementation level. Addition of a new domain concept requires making a few assertions and rerunning the automatic program writer rather than extensive manual recoding.

In addition, as in NEOMYCIN, we believe that separating different forms of knowledge will also reduce the amount of work that has to be done to move to a new domain. EES has been used to construct systems in several domains: an advice-giving system that aids users in enhancing their LISP programs, a diagnostic system that locates faults in simple electronic circuits, and a diagnostic system for identifying faults in a local area network. While we are gaining empirical evidence that the problem-solving architecture of EES is general enough to support several classes of problems, in particular advice-giving and diagnosis tasks, we are also obtaining information about the generality of the explanation facility we proposed in EES. In particular, we are gaining experience in determining how viable it is to provide a domain-independent explanation component.

8 Current Directions

The studies of conventional expert systems explanation facilities discussed in Sections 4 and 5 led to several important observations. First, inadequacies in the knowledge bases of early systems were identified and led to knowledge bases that separately and explicitly represent the types of knowledge needed to support explanations. These included justifications of the systems' actions, explications of general problem-solving strategies, and definitions of terminology. Second, we have realized that explanation is a problem in its own right, requiring its own expertise and a sophisticated problem-solving architecture. The improvements that came simply by improving expert system knowledge bases were not sufficient. In fact, the knowledge bases of newer expert systems which separate different types of knowledge [Swartout and Smoliar, 1987b] and represent knowledge at various levels of abstraction [Patil, 1981], confront explanation generators with an array of choices about what information to include in an explanation and how to present that information that were unavailable in the simple knowledge bases of earlier systems. Meeting the challenge posed by these richer knowledge bases will require more sophisticated and linguistically motivated explanation generators. Further, we now understand that explanation cannot be an afterthought, it must be *designed into* the system from the outset.

We have seen the emergence of explainable expert system frameworks, such as EES and (possibly) NEOMYCIN, that provide system builders with the tools they need to develop systems that will be explainable. Experience with these shells has shown that building an explainable system requires more work during system development, but that the discipline imposed by explanation requirements improves the architecture for other aspects of the software life cycle, in particular, knowledge acquisition and system maintenance.

The most promising course for the future is to provide system developers with an explainable intelligent system shell that can be customized for specific application domains. The shell would provide a domain-independent knowledge base, weak methods for problem solving, domain-independent explanation strategies, a lexicon for closed-class words, and user model acquisition facilities. This shell could also be augmented with tools for adding domain-specific knowledge, such as editors, authoring tools, browsing tools, and domain lexicons.

While the feasibility of this approach can only be verified empirically as more systems are developed using shells such as the EES framework, the community has learned much that can be useful in the assessment of explanation systems by identifying the constraints that the need to provide explanations places on the knowledge representation and reasoning processes of an intelligent system. These requirements themselves provide a set of metrics that can be used to determine whether a system can readily accept an explanation module.

Acknowledgements

The research on the Explainable Expert Systems framework described in this paper was supported by the Defense Advanced Research Projects Agency under Grant MDA 903-81-C-0335 and under a NASA Ames cooperative agreement number NCC2-520. I would like to thank Sean McLinden and Alan Lesgold for helpful comments on earlier drafts of this paper.

References

[Appelt, 1981] Douglas E. Appelt. *Planning Natural Language Utterances to Satisfy Multiple Goals.* PhD thesis, Stanford University, 1981.

[Appelt, 1985] Douglas E. Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, England, 1985.

[Buchanan and Shortliffe, 1984] Bruce G. Buchanan and Edward H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley Publishing Company, 1984.

[Cawsey, 1989] Alison Cawsey. *Generating Explanatory Discourse: A Plan-based Interactive Approach.* PhD thesis, University of Edinburgh, 1989.

[Chin, 1989] David N. Chin. KNOME: Modeling What the User Knows in UC. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems.* Springer-Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo, 1989.

[Clancey and Letsinger, 1981] William J. Clancey and Reed Letsinger. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. In *Proceedings* *of the Seventh International Joint Conference on Artificial Intelligence*, pages 829–836, Vancouver, B. C., Canada, 1981.

[Clancey, 1983a] William J. Clancey. The advantages of abstract control knowledge in expert system design. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 74–78, Washington, D.C., August 22-26, 1983.

[Clancey, 1983b] William J. Clancey. The epistemology of a rule-based expert system: a framework for explanation. *Artificial Intelligence*, 20(3):215–251, 1983.

[Davis, 1976] Randall Davis. *Applications of Meta-level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases.* PhD thesis, Stanford University, 1976.

[Davis, 1980] Randall Davis. Meta-rules: reasoning about control. *Artificial Intelligence*, 15:179–222, 1980.

[Finin *et al.*, 1986] Timothy W. Finin, Aravind K. Joshi, and Bonnie Lynn Webber. Natural language interactions with artificial experts. *Proceedings of the IEEE*, 74(7), July 1986.

[Fischer, 1987] Gerhard Fischer. A critic for LISP. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 177–184, Milan, Italy, August 1987.

[Hasling *et al.*, 1984] Diane W. Hasling, William J. Clancey, and Glenn Rennels. Strategic explanations for a diagnostic consultation system. *International Journal of Man-Machine Studies*, 20(1):3–19, January 1984.

[Kass and Finin, 1988] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, September 1988.

[Kass, 1988] Robert Kass. *Acquiring a Model of the User's Beliefs From a Cooperative Advisory Dialogue.* PhD thesis, University of Pennsylvania, 1988. Published by University of Pennsylvania as Technical Report MS-CIS-88-104.

[Kobsa, 1989] Alfred Kobsa. A taxonomy of beliefs and goals for user models in dialog systems. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems.* Springer-Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo, 1989.

[Kobsa, 1990] Alfred Kobsa. Modeling the user's conceptual knowledge in BGP-MS, a user modeling shell system. In *Proceedings of the Second International Workshop on User Modeling.* AAAI and the University of Hawaii, 1990.

[McCoy, 1985] Kathleen F. McCoy. *Correcting Object-Related Misconceptions.* PhD thesis, University of Pennsylvania, December 1985. Published by University of Pennsylvania as Technical Report MS-CIS-85-57.

[McKeown, 1982] Kathleen R. McKeown. *Generating Natural Language Text in Response to Questions About Database Structure.* PhD thesis, University of Pennsylvania, 1982. Published by University of Pennsylvania as Technical Report MS-CIS-82-5.

[McKeown, 1988] Kathleen R. McKeown. Generating goal-oriented explanations. *International Journal of Expert Systems*, 1(4):377–395, 1988.

[Moore and Paris, 1989] Johanna D. Moore and Cécile L. Paris. Planning text for advisory dialogues. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics*, Vancouver, B.C., Canada, June 26-29 1989.

[Moore and Swartout, 1989] Johanna D. Moore and William R. Swartout. A reactive approach to explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 20-25 1989.

[Moore and Swartout, 1990] Johanna D. Moore and William R. Swartout. Pointing: A way toward explanation dialogue. In *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, July 29 - August 3 1990.

[Moore, 1989] Johanna D. Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems.* PhD thesis, University of California, Los Angeles, 1989.

[Neches *et al.*, 1985] Robert Neches, William R. Swartout, and Johanna D. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering*, SE-11(11), November 1985.

[Paris, 1988] Cécile L. Paris. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics*, 14(3):64–78, September 1988.

[Paris, 1990] Cécile L. Paris. Generation and explanation: Building an explanation facility for the explainable expert systems framework. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers, Norwell, MA, 1990.

[Patil, 1981] Ramesh S. Patil. *Causal Representation of Patient Illness for Electrolyte and Acid-base Diagnosis.* PhD thesis, Massachusetts Institute of Technology, October 1981. Published by Massachusetts Institute of Technlogy as Technical Report Number MIT/LCS/TR-267.

[Pavlin and Corkill, 1984] Jasmina Pavlin and Daniel D. Corkill. Selective abstraction of AI system activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 264–268, Austin, Texas, August 6-10 1984.

[Pollack *et al.*, 1982] Martha E. Pollack, Julia Hirschberg, and Bonnie Lynn Webber. User participation in the reasoning processes of expert systems. In *Proceedings of the Second National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, August 18-20 1982. A longer version of this paper is available as a Technical Report from the University of Pennsylvania, Report Number CIS-82-10.

[Reichman, 1981] Rachel Reichman. *Plain-speaking: A Theory and Grammar of Spontaneous Discourse.* PhD thesis, Harvard University, Department of Computer Science, Cambridge, Mass., 1981.

[Rich, 1989] Elaine Rich. Stereotypes and user modelling. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems.* Springer-Verlag, Symbolic Computation Series, Berlin Heidelberg New York Tokyo, 1989.

[Ringle and Bruce, 1981] Martin H. Ringle and Bertram C. Bruce. Conversation failure. In Wendy G. Lehnert and Martin H. Ringle, editors, *Knowledge Representation and Natural Language Processing*, pages 203–221. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1981.

[Robinson, 1984] Jane J. Robinson. Extending grammars to new domains. Technical Report ISI/RR-83-123, USC/Information Sciences Institute, January 1984.

[Rubinoff, 1985] Robert Rubinoff. Explaining concepts in expert systems: The CLEAR system. Technical Report MS-CIS-85-06, University of Pennsylvania, 1985.

[Shortliffe, 1976] Edward H. Shortliffe. *Computer Based Medical Consultations: MYCIN.* Elsevier North Holland Inc., 1976.

[Sidner, 1979] Candace L. Sidner. *Toward a Computational Theory of Definite Anaphora Comprehension in English Discourse.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1979.

[Sparck Jones, 1984] Karen Sparck Jones. User models and expert systems. Technical Report No. 61, University of Cambridge Computer Laboratory, December 1984.

[Suchman, 1987] Lucy A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press, Cambridge, England, 1987.

[Swartout and Smoliar, 1987a] William R. Swartout and Stephen W. Smoliar. Explaining the link between causal reasoning and expert behavior. In *Proceedings of the Symposium on Computer Applications in Medical Care*, Washington, D. C., November 1987. (also to appear in "Topics in Medical Artificial Intelligence"; Miller, P.L. (ed), Springer-Verlag).

[Swartout and Smoliar, 1987b] William R. Swartout and Stephen W. Smoliar. On making expert systems more like experts. *Expert Systems*, 4(3), August 1987.

[Swartout, 1983] William R. Swartout. XPLAIN: A system for creating and explaining expert consulting systems. *Artificial Intelligence*, 21(3):285–325, September 1983. Also available as ISI/RS-83-4.

[Swartout, 1990] William R. Swartout. Evaluation criteria for expert system explanation, July 29 1990. Presented at the AAAI90 Workshop on Evaluation of Natural Language Generation Systems, Boston, Massachusetts.

[Teach and Shortliffe, 1984] Randy L. Teach and Edward H. Shortliffe. An analysis of physicians' attitudes. In *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 34, pages 635–652. Addison-Wesley Publishing Company, 1984.

[van Beek, 1986] Peter van Beek. A model for user specific explanations from expert systems. Technical Report CS-86-42, University of Waterloo, 1986.

[Webber and Joshi, 1982] Bonnie Lynn Webber and Aravind Joshi. Taking the initiative in natural language data base interactions: Justifying why. Technical Report MS-CIS-82-1, University of Pennsylvania, 1982.

[Weiner, 1980] J. L. Weiner. BLAH, a system which explains its reasoning. *Artificial Intelligence*, 15:19–48, 1980.

[Wick and Thompson, 1989] Michael Wick and William B. Thompson. Reconstructive explanation: Explanation as complex problem solving. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 20-25 1989.

[Wolz, 1990] Ursula Wolz. The impact of user modeling on text generation in task-oriented settings. In *Proceedings of the Second International Workshop on User Modeling*. AAAI and the University of Hawaii, 1990.

[Young, 1987] Young. Using the full power of computers to learn the full power of computers. Technical Report 87-9, Institute of Cognitive Science, University of Colorado, Boulder, CO, 1987.

Assessment of Software Engineering

Kathleen M. Swigger
University of North Texas

ABSTRACT

The purpose of this paper is to a) review current trends and experimental results which have immediate application in software engineering and b) offer a model of human behavior that may be useful for other types of research (e.g. educational technology). In its broadest sense, software engineering encompasses all human uses of computers but concentrates on software development. In order to improve software development, a wide range of methods have been proposed for every phase of the development cycle, from cost estimation to design, from implementation to product integration. In spite of these efforts, there appears no satisfactory method for assessing program quality or programmer productivity other than counting the number of lines of code; a measure that encourages programmers to produce lengthy rather than lucid code.

This paper reviews some of the major techniques that have been developed for assessing software production tasks. It also reports on several experimental studies that attempt to assess different software practices. The paper also discusses some of the implications of software engineering related to issues of technology assessment. Finally, the paper concludes with some suggestions for alternatives to software engineering assessment that reflect the human behavior aspects of software development.

I. INTRODUCTION

In 1967, a NATO study group was formed to discuss the "software crises." At the end of one year, the group concluded that building software is similar to other engineering tasks and that software development should be viewed as an engineering-like activity. Thus, the phrase "software engineering" was born, along with the belief that programming was simply the application of certain scientific and engineering principles. As a result, texts were written and metrics established for the purpose of identifying the scientific principles of software engineering (Gelperin & Hetzel, 1988). The fact that programs still contain bugs, are delivered late, and are over budget, should tell us that many of the basic scientific principles of programming remain undiscovered. Yet the goal remains that software engineering is a discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements.

In order to meet these goals, the scope of software engineering has become extremely broad, encompassing every phase of the software life cycle, from requirements to decommissioning. It also includes different aspects of human knowledge such as economics, social science, and psychology. To this end, a variety of techniques have been developed for performing and evaluating various software production tasks, from requirements and specifications to maintenance. In addition to measuring the quality of software, there are numerous studies that compare different techniques and methodologies used to write, comprehend and debug software. As a result, the relatively new challenge for software engineers is to develop assessment techniques that work and possibly reflect the more human aspects of software development, those that acknowledge the importance of both the programmer and the user.

The purpose of this chapter is to review some of the major techniques that have been developed for assessing software production tasks and to show how these models might be applied to other areas interested in technology assessment. To this end, this chapter has three major themes. The first is analysis and comparison. A variety of techniques are described for software evaluation and the evaluation of software engineering. Software evaluation is defined as the assessment of a specific piece of code or program produced by individuals and/or a team of programmers. On the other hand, software engineering refers to the practices, techniques, and procedures that are used to produce correct and quality software. Because of the plethora of present-day software engineering techniques, it is important to select an appropriate one for the task at hand. A second theme is that the results of experiments in software evaluation and software engineering constitute a powerful tool for determining which techniques are useful for a given situation. These same techniques and tools might be used by other researchers in related fields of technology. The third theme is that the future of software production will necessitate the development of a new definition of software engineering that recognizes the human aspects of both software development and its evaluation. Unfortunately, the human element is a factor in both the design and use of software.

In order to develop these themes, the chapter focuses on different perspectives of software engineering. For example, the next section introduces the software evaluation perspective. The wide scope of different measures used to evaluate software is highlighted, as are the problems of conducting software evaluation tests. Section III compares and contrasts different software engineering methodologies. Experiments related to the different methodologies are also reported. Section IV includes a discussion of current software engineering topics such as Computer Aided Software Engineering (CASE) products and Computer Supported Cooperative Work environments (CSCW) and shows how each of these topics reflect an understanding of human problem solving. These topics were selected because of their relevance both to current software engineering practices and to the theme of this book. The chapter concludes with a brief summary and discussion of future research.

II. SOFTWARE EVALUATION

II. A. Definition and Characteristics

Since the 1970s the distinction between software and software engineering has become blurred. In the "good old days" the distinction was very clear. Software was a single piece of code that was compiled and executed. On the other hand, software engineering referred to the set of systematic procedures that were applied and used when developing a collection of programs (Schach, 1990). Now, however, the creation of an isolated piece of code is extremely rare. Most software is written by teams of programmers which access and control several other programs or hardware. However, in order to distinguish among the different software evaluation techniques, the rest of the chapter returns to the "good old days" and uses the term "software" to denote the end result of a process (i.e., a product), whereas software engineering refers to the process of developing the software. Thus, software evaluation involves the process of ensuring that the actual product itself is "correct," while evaluation of software engineering involves testing whether the procedures used to produce the software are correct.

Software quality measurement is a young discipline. Because of its relative youth, there are conflicting opinions as to what and how specific software characteristics should be measured. For example, several authors have argued that software evaluation is the testing of a program until it no longer contains any bugs. Unfortunately, as Dijkstra points out, "Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence" (Dijkstra, 1976). Software evaluation, therefore, implies the testing of a product to determine if it is correct (program verification) and exhibits certain behavioral properties (program validation).

The primary goal of any testing procedure is to determine whether the product functions correctly. Additional software characteristics include utility, reliability, robustness, performance, and correctness (Gelperin & Hetzel, 1988). Utility refers to the extent to which a program meets the user's needs, given that the product is used according to its specifications. The utility of the product is determined by verifying that the program produces correct outputs when subjected to inputs that are valid in terms of its specifications.

Reliability is a measure of the frequency and criticality of product failure, where failure is an unacceptable effect or behavior occurring under permissible operating conditions. Software reliability is calculated by adding the mean time between failures to the mean time for repairing the failures.

A product's robustness is a function of the range of operating conditions, the possibility of unacceptable effects on valid input, and the acceptability of effects when the product is given invalid input. A fourth characteristic, performance, is defined as the extent to which the product meets its constraints with regard to response time, execution times, or space requirements.

The last characteristic, correctness, refers to a mathematical procedure that is used to prove that a product satisfies its output specifications, when operated under permitted conditions (Goodenough, 1979). In other words, the product is correct if the output satisfies the output specifications, given that the product has all necessary resources.

Having identified the important software characteristics, it becomes necessary to construct appropriate tests that measure each of the different characteristics (Figure 1). Unfortunately, different types of software emphasize different software characteristics. For example, programs using artificial intelligence techniques emphasize performance criteria rather than accuracy or optimality (Brazile & Swigger, 1990). Indeed, speed of execution is often the major reason for selecting an AI technique over other conventional methods. In contrast, an operations research technique is designed to produce optimal results at the expense of high performance. Because of space limitations, it is impossible to list all the testing procedures that are used in combination with the various types of software. The remainder of the section, therefore, describes only a few testing techniques that are used by program developers. The reader is encouraged to investigate other sources for additional information (Gelperin & Hetzel, 1988; Perry, 1983).

Figure 1 goes about here

I. B. Walkthroughs and Code Inspections

A Walkthrough is a software review performed by a team of software professionals with a broad range of skills (Shneiderman, 1980). The team is usually comprised of four to six members who are charged with the task of offering an unbiased report of the software under construction. Thus, the lead programmer "walks" the other members through the program. The members interrupt either with prepared comments or with questions triggered by the presentation. In this manner, the software is examined for faults or irregularities.

A Code Inspection is another form of software review. The team reviews the program specifications against a prepared checklist which includes items such as: Have the hardware resources required been specified? Have the acceptance criteria been specified? A Code Inspections is a more formalized review process and usually involves five basic steps (Fagan, 1976). First, an overview of the design is given by the lead programmer. Second, the code inspection team prepares comments individually for the inspection. Third, the group inspects the code which involves addressing each of the individual comments and ensuring that every piece of logic is covered at least once, and every branch is taken at least once. The fourth step, called rework, involves resolving all faults and problems. The final stage involves a follow-up in which the inspection team ensures that all questions have been satisfactorily resolved.

I. C. Selection and Use of Test Cases

In order to verify that the program functions correctly, a group of test cases is constructed to either test to specifications (also called black-box, data-driven, functional testing), or test to code (also called glass-box, logic-driven, or path-oriented testing). Using the former technique, the programmer constructs a series of test cases that correspond to the software's specifications (Perry, 1983). In contrast, the test to code technique requires the programmer to construct test cases that consider only the code itself. Regardless of which technique is selected, a "complete" testing program requires the construction of literally billions and billions of test items. Therefore, the "art" of test case construction is to find a small, manageable set of test cases that maximize the chances of detecting a fault while minimizing the chances of having the same fault detected by more than one test case (Myers, 1978).

I. C. 1. Testing to Specifications

Equivalence Testing and Boundary Value Analysis

To determine if the product runs correctly the program designer constructs a set of test cases such that any single member of the class is representative of (or equivalent to) all other members of the class (Schach, 1990). For example, if a program is written to handle a range of numbers between 1 and 35, then the programmer defines three different equivalence classes:

Equivalence class 1: numbers less than 1.
Equivalence class 2: numbers between 1 and 35.
Equivalence class 3: numbers more than 35

Testing a program using the equivalence class technique requires constructing one test case from each equivalence class. The test case from equivalence class 2 would produce the correct answer, while the test cases from the other two classes would produce error messages.

Experience has shown that, when a test case is selected from either side of the boundary of an equivalence class, there is a high probability of locating a fault. Testing the previous example using this technique (i.e., boundary value analysis) would produce seven different test cases:

Test case 1: 0 (number adjacent to boundary condition)
Test case 2: 1 (boundary value)
Test case 3: 2 (number adjacent to boundary condition)
Test case 4: 15 (member of equivalence class 2)
Test case 5: 35 (boundary value)
Test case 6: 36 (number adjacent to boundary condition)
Test case 7: 37 (number adjacent to boundary condition)

Equivalence class testing combined with boundary value analysis is an effective technique for locating faults and requires a relatively small set of test data. Research has shown that, when used together, these two methods constitute an extremely powerful evaluation tool (Basili & Selby, 1987).

Functional Testing

An alternative to the above technique is to construct test cases based on a program's functionality (Howden, 1987). After defining the program's functions, test data are then constructed such that there is at least one test case for every function in each module in the program. If the modules are designed in a hierarchical fashion, than functional testing proceeds in a bottom-up manner. In practice, however, modules and subroutines are highly interconnected and require complex functional testing techniques; for details, see Howden, 1987.

II. C. 2. Testing to Code

Structural Testing: Statement, Branch, and Path Coverage

The simplest form of Code Testing is to examine each, individual statement (ie., statement coverage), and construct a of test case that ensures that every statement is correctly executed (Schach, 1990). Usually an automated tool is required to record which statements have been executed over a series of tests. Branch coverage is another type of functional testing and involves running a series of tests to ensure that all branches in the program are executed at least once. The most powerful form of structural testing is path coverage which requires testing all paths through the program. Unfortunately, if the product contains many loops, the number of paths through a program can be computationally quite large. Thus, the programmer learns to reduce the number of paths by restricting test cases to only linear code sequences (Woodward, Hedley, & Hennell, 1980), or to code sequences that lie between the declaration of a variable and its use (Rapps & Weyuker, 1985).

II. D. Complexity

Software complexity is an attempt to merge theories from cognitive psychology with theories from computer science. Similar to psychological measures, software complexity refers to a program's characteristics that make it difficult for a human to understand. The underlying assumption behind these metrics is that a product's complexity is a good predictor for product reliability, performance, and flexibility (Shneiderman, 1980). Thus, if the complexity of a program is measured and found to be extremely high, then the module should be rewritten because it will be cheaper and faster to start all over than to try and debug and maintain the existing code.

The simpliest and most frequently used measure of complexity is lines of code. Although this type of measure has proven ineffective for determining programmer productivity, it can be a useful predictor of the number of faults in a program (Basili & Hutchens, 1983; Takahashi and Kamayachi, 1985).

Other, more accurate, predictors of product complexity and fault rates look at either the number of decision points in a program or the number of operators and operands. For example, McCabe (1976) developed a measure of complexity based on graph theory that counts the number of branches in a program plus 1. He argues that complexity of a total product consisting of N modules is the sum of the complexity of the individual modules. McCabe's metric can be computed almost as easily as lines of code and has been shown to be a good predictor for the number of faults in a program (Schach, 1990).

Halstead's Software Science metric (Halstead, 1977) has also been used to measure complexity and fault rates. Halstead's method is based upon the ability to count, for any program, the number of unique operators (such as IF, =, DO, PRINT) and the number of unique operands (such as variables or constants). The other two basic elements are the total number of operators and the total number of operands. From those counts, Halstead derived functions for predicting properties such as program length, program volume, and program level. Subsequently, he used those results with theories and assertions relative to cognitive psychology and derived equations that predict the mental effort and time required to write different programs. He also speculated about the relationship between program metrics and text analysis.

Although the idea of measuring a program's complexity is appealing, the exact nature of its use remains in question. The development of a theory of programming based on the most primitive components of programs - operators and operands - unquestionably is appealing. However, the ability to provide measures that accurately reflect and predict the mental processes involved in programming has not been fully documented.

II. E. Correctness

Recently, the idea of correctness and correctness proofs has become a major topic in computer science (Dijkstra,1990). In an an attempt to provide a mathematical framework for computer programming, several researchers have developed special verification techniques that prove the correctness of a program.

The major difference between testing and correctness proofs is that testing is performed by EXECUTING a program, while a correctness proof is a mathematical verification that the product is correct; the product is NOT executed using a computer. A program is said to be correct if its output satisfies its input specifications. This, of course, does not necessarily mean that the product is acceptable to the user. It only means that the product satisfies its specifications.

Space requirements prohibit an elaborate explanation of correctness proofs. A brief example of a code fragment along with its corresponding flow chart and proof is provided for the interested reader (Figure 2). Additional details on how to perform correctness proofs can be found in (Manna, 1974) and (Dijkstra, 1976).

Figure 2 goes about here

It has been shown that correctness proofs by themselves are insufficient to verify programs. It has been demonstrated that a program that is proved correct may still contain several errors (Leavenworth, 1970; Goodenough & Gerhart, 1975). The studies indicate that the combined use of test cases and correctness proofs is the only way to ensure that a program contains no faults. Thus, correctness proving must be viewed as belonging to the larger set of techniques that can be used to check that a product is correct.

II. F. Implications of Software Testing to Technology Assessment

A number of studies have been performed comparing different strategies for testing software. For example, Myers (1978) compared specification-testing techniques with different code-testing techniques and structured walkthroughs. Similarly, Basili and Selby (1987), compared specification testing, code testing and code inspection techniques. Both studies found that the different testing techniques were equally effective, with each technique having its own unique strengths and weaknesses. Although no one technique was found to be superior, they were all found to be better than using no technique.

The obvious implication of such studies is that educational software (as well as other types of application software) is <u>software</u> and, as such, requires both testing and verification. The application of software evaluation techniques to the evaluation of educational software should be performed at each stage of the software development process. It is interesting to note, for example, that educational researchers routinely report measures of validity and reliability for studies relating to IQ and skill acquisition. Yet, these same researchers rarely report measures of program reliability or validity for educational software. The absence of such measures seems to indicate that authors of educational software are either ignorant of software testing procedures or incompetent to perform such tests. Hopefully, this is not the case and that program reliability and validity for educational software will be reported in the near future.

Other implications of the software evaluation process is that software testing techniques, especially those used for the construction of test cases, can be applied to other areas of technology assessment. For example, in order to demonstrate that software is appropriate for different types of populations, it is sufficient to show that members from different equivalence classes can perform equally well.

Although software testing may ensure that the product contains no faults, none of the above techniques ensures that the user will like the product. Software evaluation measures simply test for errors and violations of specifications. If the specifications are poorly defined, then the software may be unacceptable to the user. As a result, the software evaluation process must consider the context in which the product is used and produced.

III. EVALUATION OF SOFTWARE ENGINEERING

III. A. The Life-Cycle Models

As previously stated, the idea that a product exists as a single piece of code is no longer valid. A program may be required to run in parallel, on different machines, under different operating systems, and accessing multiple databases. As a result, special methodologies, models, and procedures are used to systematize the production of large programs. The broad term assigned to these approaches is the "life-cycle model" because it describes procedures for carrying out the various functions of software development. Once a model has been selected, milestones are established and an overall plan for product development is established.

III. A. 1. The Waterfall Model

Two software engineering models commonly used are: the

Waterfall Model and the Rapid Prototype Model. Actually, the two models are very similar to each other and vary only in one area.

As first proposed in 1970, the Waterfall Model (Royce, 1970) describes the conventional approach to software development. The version, as it appears in Figure 3, suggests that the software development cycle consists of six separate phases: requirements, specifications, design, implementation, integration, and operations. Following each phase is a period of testing and verification which, if unsuccessful, forces the developer to reevaluate previous specifications and design. The Waterfall Model with its feedback loops and iterative design process allows for revisions of the design, and even the specifications, at every stage of the process. It should be noted that testing is not a separate phase of the process, but occurs continuously throughout the life cycle of the product.

Figure 3 goes about here

Although communication with the client occurs at each phase of the life cycle, a list of specifications does not always tell the user how the finished product will look or feel. Thus, the Waterfall Model, depending on how it is implemented, can sometimes lead to software that is unacceptable to the user. As a result, the Rapid Prototyping Model was developed to solve this problem.

III. A. 2. The Rapid Prototype Model

A prototype is a working model that is functionally equivalent to a subset of the product. The first step in the Rapid Prototyping life cycle is to specify a product's functionality and then build a program that matches those specifications. Once the client is satisfied, the software development process continues as shown in Figure 4. The two most important items to remember when using the Prototype Model is that 1) the prototype is built for change; and 2) the prototype is built as input to the specification stage. Thus, the prototype is simply a minor detour from the normal path of software development.

Figure 4 goes about here

The use of rapid prototyping as a way of minimizing risk is the idea behind the Prototype Model. Unfortunately, rapid prototypes are often accepted as the end product, or as a substitute for written specifications. Another potential problem is that prototypes may not adequately assess hardware needs for large-scale software products. There are substantial differences between large-scale and small-scale software, and a prototype cannot adequately assess the type of hardware needed for large-scale tasks.

Yet Rapid Prototyping combined with the Waterfall Model can produce an acceptable life-cycle model for developing software. Prototypes are extremely useful for demonstrating how the interface will look to the end user. On the other hand, the Waterfall Model provides a systematic set of procedures for designing, implementing, and integrating large-scale products.

III. A. 3. The Implications of the Life Cycle to Technology Assessment

As previously stated, testing is an inherent component of the entire product life cycle and requires careful validation at every stage of development. Following each phase of product development, specific tests are performed and examined. For example, a structured walkthrough is staged during the specification stage, while module testing and test case selection occurs during the implementation phase (Figure 5). The idea that different tests are used at different times during product development should be applied to other areas of technology assessment. It is not uncommon, for example, to find a research team comparing students' performance using different media. These studies consist of a single test designed to determine whether the technology meets its requirements, is integrated correctly, and outperforms all other treatments. A better approach to software evaluation is to design multiple tests in parallel with every stage of product development.

Figure 5 goes about here

III. B. The Design Phase

Rather than describe each phase of the software life cycle, together with its appropriate testing procedures, this section focuses on the DESIGN phase of software development. Just as an an outline serves as a catalyst for written works, a design technique drives effective software development. Thus, program design is a very critical phase of the software development life cycle. Techniques and tools that effectively represent requirements in a format that results in a fault-free product are essential to a good program design. A description of these tools and their evaluation, form the subject of this section.

A design methodology is an artificial language that enables the programmer to describe a particular type of problem at a conceptual, rather than implementation, level. The tools (ie., pseudo code, Warnier Orr Diagrams, Hypo-Charts) that are derived from the design methodology allow the programmer to be precise about which parts of the program are program specific and which parts address a more general design plan. Programmers who use design methodologies perform better because they are forced to divide the problem into smaller modules that are easier to design, code, and debug. Such a methodology permits the designer to cooperatively develop systems using a shared language of architecture constructs, rather than a set of problem specific primitives.

Different design methodologies tend to emphasize different aspects of the programming process. For example, some design methodologies are very effective for showing data and the relationship among data items, while other methodologies stress data flow or program functionality. Two design techniques that highlight different aspects of the design process are Petri nets and Entity-Relationship (ER) diagrams. Petri nets have proven extremely useful for describing real time systems because they describe the flow of data throughout the system. In contrast, ER diagrams are effective for representing the object-oriented programming paradigm because they show data and the relationship among data. The connection between the specification language and the problem description can be very critical as shown below.

Petri nets are abstract, formal models of information flow that look very similar to directed graphs. As illustrated in

Figure 6, nodes are used to represent completion of events, while arcs represent transitions from one event to another (Peterson, 1980). Petri nets have been successfully used for problems relating to parallel computation (Miller, 1979), multiprocessing (Agerwala, 1979), knowledge representation (Jantzen, 1980), and human information processing (Schumacker & Geiser, 1978). Although Petri nets can be used to represent descriptive data, they are more suited to describing information flow. For this reason, Petri nets are useful for specifying real-time systems, with timing issues being critical.

Figure 6 goes about here

As illustrated in Figure 7, an Entity-Relationship (ER) diagram consists of nodes that represent entities and arcs that represent the relationship between two entities. ER diagrams were first introduced by Chen (1976) who used them to describe the Entity-Relationship database model. As such, ER diagrams are more appropriate for describing data and the relationships among data. More recently, ER diagrams have been proposed as a design language for knowledge-base development (Addis, 1985; Swigger & Brazile, 1988), and have proven effective for describing object-oriented systems (Boehm-Davis, et. al., 1986).

Figure 7 goes about here

III. B. 1. Design Comparisons

Many other formal techniques have been proposed. For example Anna (Luckham & von Henke, 1985) is a formal specification language for Ada, while Refine (Smith, Kotick, &

Westfold, 1985) and Gist (Balzer, 1985) are used to describe knowledge-based systems. Research has recently discovered that the decision to use a particular design technique for a specific project depends on the problem that needs to be solved. Boehm-Davis and Ross (1984), Boehm-Davis, Holt, Schultz, and Stanley (1986), Boehm-Davis and Fregly (1983) performed a number of studies which were aimed at determining the effect of using different design/documentation formats in a variety of comprehension, coding, verification, and modification tasks. The Boehm-Davis and Ross (1984) first determined that performance on a set of software tasks was linked to documentation type. Boehm-Davis and Fregly (1983) next compared different documentation formats such as PDL, abbreviated English, and Petri nets and found that performance scores varied as a function of documentation type. Finally, Boehm-Davis et al. (1986) asked experienced programmers to modify several different types of programs in combination with several different types of design tools. The authors concluded that different design/documentation formats did indeed effect both design time and problem solution time and that the differences could be attributed to the different types of problems.

Similar experiments have examined different design tools used to program expert systems. Swigger and Brazile (1989; 1990) found that programmers who used a design tool performed significantly better on modification tasks than programmers who did not use a design tool. Results also suggested that there were differences between different types of design tools, and that different design tools effected different types of programming tasks.

What seems to be important for the development of both conventional software as well as expert systems is that the design technique provide a uniform representation and organization of the more general problem description. This should also be true of design tools that are used to develop educational software and other types of applications. Thus, it appears to be necessary to identify a software product as belonging to a specific class or type of problem and then use the design techniques that best represent the problem type. This type of classification relates to both the domain knowledge and programming techniques that are used to solve the problem.

IV. AUTOMATED PROGRAMMING TOOLS

The idea that domain knowledge and programming knowledge are equally important for the construction of good software has had a major impact on the development of automated programming tools. Although the concept of a total automatic programming environment remains a fantasy, there are several recent developments that bring the fantasy closer to reality. These types tools are known as Computer Aided Software Engineering (CASE) products.

In the past, the terms automatic programming and CASE were applied to any type of programming tool that automated any part of the program life cycle (Balzer, 1985). It is only recently that companies have developed products that, through a successive series of steps, are able to transform specifications into executable source code. The transformation process is by no means fully automated; human intervention is required in deciding which transformations to apply, and precisely where to apply the transformations. Underlying these successful products is a model of programming as well as a model of the domain.

One current CASE tool conceptualizes the programming model as consisting of inputs, processes, and outputs (Figure 8). Input objects include both screen and file objects. Output objects include screen and file objects as well as report objects. Then, depending on the application, process objects (eg., sort, sequence, retrieve, etc.) are used to transform the input/output objects. The CASE tool creates the different objects (ie., screen, report, sort, etc. objects) by asking the programmer to supply both domain-specific and programming knowledge. For example, the CASE tool creates a specific report object by asking the programmer to provide the format of the report, the heading, the names of the specific data items to be processed, the primary control break, etc. Thus far, CASE tools have been developed only for restrictive domains such as business applications (Frenkel, 1985), database problems (Kaiser, Feiler, & Popovich, 1988), and data analysis problems (Balzer, 1985).

Figure 8 goes about here

Another approach to automated programming is to build specialized tools for a specific programming language. Such tools can handle much of the drudge work of programming, leaving the creative work to the human programmer. Powerful debuggers, intelligent editors, and elaborate programming environments are included under this approach. It has been argued that the use of such tools is, in itself, a sufficient condition for increased programmer productivity and effectiveness (Barstow, 1985). For example, there are powerful debuggers for the C programming language, that optimize code, provide powerful debug messages, and suggest effective programming styles.

A third type of computer tool focuses on the problem of supporting team programming and large product development projects. Computer-supported cooperative work (CSCW) is a research area that includes the development of computer systems that support group design activities. For example the Software Technology program at Microelectronics and Computer Technology Consortium is working on the problem of Issue Based Information Systems (IBIS) which will help software designers by supporting structured collective conversations through planning (Conklin, 1986). In a similar manner, this author has recently built and evaluated an intelligent interface to support computer-supported cooperative problem solving. The system serves as a testbed for investigating tools that people use while engaged in technical, cooperative tasks such as working on large programming projects (Swigger et. al, 1990). Each of the online tools represents and is linked to a requirement for successful communication (Figure 9). For example, following an examination of how to build common vocabularies, an online tool was created that enhances this requirement. Thus, the system is designed to test a "theory of communication" which states that effective cooperative problem solving is dependent on effective communication which, in turn, consists of common vocabularies, syntax, objectives, etc. If the communication model is correct, then the online tools should enhance communication performance.

Figure 9 goes about here

Whether in the form of a CASE tool, a powerful debugger, or a CSCW tool recent advances in programming incorporate models of programming as well as models of the domain. As research indicates (Pennington, 1987), computer programming is a highly complex task with many components. It is often compared to other types of tasks in an attempt to understand its underlying processes. For example, programs have been compared to texts (Atwood & Ramsey, 1978). As such, they are described as having organization and structure, and programmers are said to have general schemata that guide encoding, representation, and retrieval of the program-as-text (eg., Rumelhart, 1981). Programming has also been compared to expert skills such as playing chess or Go, and diagnosing faulty electric circuits. This particular analogy focuses attention on the potentially large stores of specific programming patterns that have been learned through extended practice (Chase & Simon, 1973; Chi, et al., 1981; Barstow, et. al., 1984). Finally, programming has been analyzed as a planning and problem solving task that utilizes some general strategies such as problem decomposition and reformulation (Newell & Simon, 1972); Miller & Goldstein, 1979); Soloway, et.al., 1988) The suggestions for program planning have been much closer to the computer scientist's view of orderly, top-down structured programming than the psychological literature would lead us to expect.

It should be noted that the above analogies are not necessarily contradictory. Programming-as-text focuses on comprehension and memory and working backwards from program to interpretation. Programming-as-planning focuses on successive construction and working forward. Programming-as-expert-skill focuses on the organization of knowledge specific to the programming domain that is clearly implicated in both comprehension and construction of programs. The challenge is to develop programming tools and software evaluation techniques that reflect the human aspects of programming as well as the specific problem domain. As such, the software tools and the software evaluation techniques need to incorporate models of programming and models of the domain.

V. SUMMARY AND IMPLICATIONS FOR FURTHER RESEARCH

One goal in extensively reviewing the existing studies on software evaluation and evaluation of software engineering has been to identify important themes that pertain to programming as well as technology assessment. Across evaluation tasks, a recurring question concerned the existence and nature of meaningful measures with which to evaluate software and software engineering practices. Several measures were examined, and a wide variety of evaluation techniques and formulations were found that address different aspects of the software development cycle.

The paper first distinguished between software and software engineering and stated that the difference related to their use and function: specific code that solves a domain specific problem versus a model or methodology for general program development. As a result of this distinction, it was possible to discuss code evaluation as opposed to the evaluation of a programming methodology. A second distinction concerned the model of programming that each of these two areas measures; error free code versus general problem solving. Although such distinctions exist, it was noted that most programs are now written as part of larger systems. As such, software and software engineering involve the use of general problem solving strategies as well as specific domain knowledge. Thus, a second recurring set of questions concerned the definition of software (or software engineering) and, indirectly, the definition of programming: (1) Is programming a series of successive transformations of the external problem domain into a representation in the programming language? (2) Can a program be evaluated separate from either its domain or programming techniques? 3) Are there fundamental structural components of programming that exist?

This review also documented that conflicting evidence exists on software and software engineering evaluation. Although software evaluation studies have not demonstrated the existence of a single set of principles for software evaluation, there are several software measures that can be applied to other areas of technology assessment. Similarly, software engineering studies addressing larger issues of software design fail to report a single design methodology that results in the production of quality software over time and for every type of application. Yet, the idea of using different tests for different phases of the software development life cycle is a major lesson to be learned from these studies. A second lesson seems to be that every development group must decide what type of evaluation procedure is appropriate for a particular problem type.

In considering the various topics in this chapter, it has been noted that it is difficult to draw definitive lines between evaluation of software and evaluation of software engineering. It has been argued that an understanding of a model of programming is closely related to both the construction of effective software and a programmer's systematic approach to a problem. A model of programming involves the cognitive representation of particular programs at the surface level of the code, at a deeper structural level, and at an interpretive level. It also involves a similar representation of the knowledge of the application and a deep understanding of how it will be used by the client.

Some reasons for focusing on developing models of programming and then using these models to derive performance measures for software is that it has implications for the development of programming aids. One can imagine, for example, a programming tool that is capable of transforming different representations at different levels of abstraction. The tool would be able to analyze domain specific information and use this knowledge to suggest possible programming strategies. A different scenario would entail using a programming tool to design screens, write procedures, and develop data structures. The tool would interrupt the programmer only when it identified an error. A third type of tool would enable programmers located in different cities and countries to work together on a single programming project. Such a tool would allow team members to exchange code, documents, and design information in an effort to create a useful, robust, reliable, and correct program. Regardless of which version of the future one prefers, the development of an effective programming tool depends on a clear understanding of the programming process as well as the application domain.

Other reasons for studying models is that that have implications for the development of measures of program assessment. Current programming measures are inadequate to evaluate large programming projects. Testing all paths or constructing sufficient test cases are impractical for current software development projects. Although every piece of software must produce correct results, it must also be acceptable to the user. Therefore, it is important to consider the human aspects of software development to determine issues of complexity, maintainability and usability. It has been documented that programmer productivity increases, complexity decreases, and program performance increases, when software engineers use "good" programming practices. Being able to define "good" programming practices in terms of a model of human performance should also improve productivity and performance.

There is a final lesson in this analysis that has relevance to the general issue of technology assessment. Once software is transferred from the programmer to the user, the question of software evaluation or even evaluation of software engineering is no longer relevant. At this point, the question should be whether the underlying "model" of pedagogy, communication, explanation, learning, etc., as represented in the computer program, is correct and effective? As a program advances to its final stages of development, it ceases to be a program and becomes a model of human performance. Therefore, product testing and evaluation should concentrate on the model and not the program.

The intention of this chapter has been to review the existing practices in software engineering for program evaluation, to identify some recurring questions, and to suggest some implications of human behavior for software evaluation and for technology assessment in general. There are clearly other avenues of research that might be pursued productively in the study of assessment of software practices. However, these avenues will necessitate am understanding of the human aspects of problem solving and the way that these aspects interact with specific domain.

References:

Addis, T.R. (1985). Designing Knowledge-Based Systems. Englewood Cliffs, NJ: Prentice-Hall.

Agerwala, T. (1979). Putting petri nets to work. Computer, December, 85-94.

Atwood, M.E. and Ramsey, H.R. (1978). Cognitive structure in the comprehension and memory of computer programs: An investigation of computer program debugging. ARI Technical Report TR-78-A210, Science Applications, Englewood, Colorado, 1978.

Balzer, R. (1985). A 15 Year perspective on automatic programming. IEEE Transactions on Software Engineering, SE-11, 1257-1268.

Barstow, D.R., Shrobe, H.E. & Sandwall, E. (1984) (Eds.), Interactive Programming Environments. New York: McGraw-Hill.

Basili, B.R. & Selby, R.W. (1987). Comparing the effectiveness of software testing strategies. IEEE Transactions of Software Engineering, SE-13, 1278-1296.

Basili, B.R. & Hutchens, D.H. (1983). An empirical study of a syntactic complexity family. IEEE Transactions on Software Engineering, SE-9, 664-672.

Boehm-Davis, D.A. & Ross, L. (1984). Approaches to structuring the software development process. (Technical Report Tr-84-Blv-1), Arlington, Virginia: General Electric Company.

Boehm-Davis, D.A., Holt, R. W., Shultz, A.C. & Stanley, P. (1986). The Role of Program Structure in Software Maintenance. (Technical Report Tr-86-gmu-pol), Fairfax, Virginia: George Mason University.

Boehm-Davis, D. A. & Fregly, A. (1983). Documentation of concurrent programs. In A. Janda, (Eds.), CHI '83 Conference Proceedings, Boston, MA., 256-261.

Brazile, R. & Swigger, K.M. (1990). Criteria for selecting different modeling techniques, Proceedings of the International Conference on Modelling and Simulation, Lugano, Switzerland.

Chase W.G. & Simon, H.A. (1973). The mind's eye in chess. In W.G. Chase (Ed.). Visual information processing, NY: Academic Press.

Chen, P. (1976). The entity-relationship model - toward a unified view of data. ACM TODS 1, 9-36.

Chi, M.T.H., Feltovich, P.J. & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-152.

Conklin, J. (1986). A theory and tool for coordination of design conversations. Tech. Rep., STP-236-86, MCC, Microelectronics and Computer Technology Consortium, Austin, Tx.

Dijkstra, E.W. (1976). A discipline of programming. Englewood Cliffs, NH: Prentice-Hall.

Fagan, M.E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 15, 182-211.

Frenkel, K.A. (1985). Toward automating the software-development Cycle. Communications of the ACM, 28, 578-589.

Gelperin, D. & Hetzel, B. (1988). The growth of software testing. Communications of the ACM, 32, 687-695.

Goodenough, J.B. (1979). A survey of program testing issues. In P. Wenger (Ed.), Directions in Software Technology. (pp. 319-340), Cambridge, MA.: MIT Press.

Goodenough , J.B. & Gerhart, S.L. (1975). Toward a theory of test data selection. IEEE Transactions on Software Engineering, SE-1, 156-173.

Halstead, M.H. (1977). Elements of software science. New York: Elsevier North-Holland.

Howden, W.E. (1987). Functional Program Testing and Analysis. New York: McGraw-Hill.

Jantzen, M. (1980). Structured representation of knowledge by Petri nets as an aid for teaching and research. Lecture Notes in Computer Science. Berlin: Springer-Verlag.

Kaiser, G.E., Feiler, P.H. & Popovich, S.S. (1988). Intelligent assistance for software development and maintenance. IEEE Software 5, 40-49.

Leavenworth, B. (1970). Review #19420, Computing Reviews, 11, 396-397.

Luckham, D.C. & von Henke, F.W., (1985). An overview of Anna, a Specification language for ADA. IEEE Software, 2, 9-22.

Manna A. (1974). Mathematical Theory of Computation. New York: McGraw-Hill.

McCabe, T.J. (1976). A complexity measure. IEEE Transactions on Software Engineering, SE-2, 308-320.

Miller, M. & Goldstein, I. (1977). Structured planning and debugging. In Proceedings of the fifth international joint conference on Artificial Intelligence, Cambridge, Mass.

Miller, P. (1979). A methodology for the design and implementation of communication protocols. IEEE Transactions on Communication, 24, 614-621.

Myers G. J. (1978). A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM, 21, 760-768.

Newell, A. and Simon, H. (1972). Human problem solving. New York: Prentice-Hall.

Pennington, N. (1987). Comprehension strategies in programming. In G. Olsen and E. Soloway (Eds.), Empirical Studies of Programming, Norwood, NJ: Ablex.

Perry, W.W. (1983). A structured approach to systems testing. Englewood Cliffs, NJ: Prentice-Hall.

Peterson, J. L. (1980). Petri Net Theory and the Modeling of Systems. Englewood Cliff, NJ: Prentice-Hall.

Rapps S., & Weyuker, E.J. (1985). Selecting software test data using data flow information. IEEE Transactions of Software Engineering, SE-11, 367-375.

Royce, W. W. (1970). Managing the development of large software systems: Concepts and techniques, Proceedings of WestCon, August, 1970.

Rumelhart, D.E. (1981). Understanding and summarizing brief stories. In D. LaBerge and S.J. Samuels (Eds.), Basic processing in reading: Perception and comprehension. Hillsdale, NJ: Erlbaum.

Schach, S.R. (1990). Software Engineering. Homewood, Ill: Aksen Associates Incorp.

Schumacker, W. & Geiser, G. (1978). Petri nets as a modeling tool for discrete concurrent tasks of the human operator. In Fourteenth Annual Conference on Manual Control, NASA Conference Publication, 2060, 161-197.

Shneiderman, B. (1980). Software Psychology: Human Factors in Computer and Information Systems. Cambridge, MA.: Winthrop Publishers.

Smith, D.R., Kotik, G.B., & Wesfold, S.J. (1985). Research on knowledge-based software environments at the Kestrel Institute, IEEE Transactions of Software Engineering, 11, 1278-1295.

Soloway, E., Pinto, J., Letovsky, S., Littman, D. & Lampert, R. (1988). Designing documentation to compensate for delocalized plans. Communications of the ACM, 31, 1259-1267.

Swigger, K. M. & Brazile, R. (1988). An architecture for an intelligent tutoring system based on entity-relationship data model. In C. Frasson (Ed.) Proceedings of the International Conferences on Intelligent Tutoring Systems, (pp. 258-263), Montreal, Canada.

Swigger, K. M. & Brazile, R. (1989). Experimental comparison of design/documentation formats for expert systems. International Journal of Man-Machine Studies, 31, 47-60.

Swigger, K.M. & Brazile, R. (1990). An empirical study of the effects of design/documentation formats on expert system modifiability. (in press).

Swigger, K.M., Swigger, B.K., Thomas, T., & Brown, S. (1990). An empirical study of computer cooperative tools for information exchange (in press).

Takahashi, M. & Kamayachi, Y. (1985). An empirical study of a model for program error prediction. Proceedings of the eighth International Conference on Software Engineering, London, UK, pp. 330-336.

Woodward, M.R., Hedley, D. and Hennell, M.A. (1980). Experience with path analysis and testing of programs. IEEE Transactions on Software Engineering, SE-6, 278-286.

UTILITY
* Usability Tests

RELIABILITY
* Walkthroughs
* Code Inspection
* Number of Bugs + Downtime between Failures

ROBUSTNESS
* Test Case Selection
* Functional Testing

PERFORMANCE
* Execution Time
* Average Response Time
* Memory Constraints
* Hardware Constraints
* Portability

CORRECTNESS
* Proofs of Correctness

Figure 1: Measurement Techniques for Program Characteristics

TESTING DURING REQUIREMENTS PHASE
* Prototype Development

TESTING DURING THE SPECIFICATIONS PHASE
* Structured Walkthroughs

TESTING DURING THE DESIGN PHASE
* Design Inspections

TESTING DURING THE IMPLEMENTATION PHASE
* Equivalence Testing & Boundary Analysis
* Functional Testing
* Statement, Branch, and Path Coverage
* Complexity
* Correctness

TESTING DURING THE INTEGRATION PHASE
* Module Testing
* Acceptance Testing
* Hardware Testing

TESTING DURING THE OPERATIONS PHASE
* Corrective Testing
* Regression Testing

Figure 5: Testing and the Life-Cycle Model

CORRECTNESS

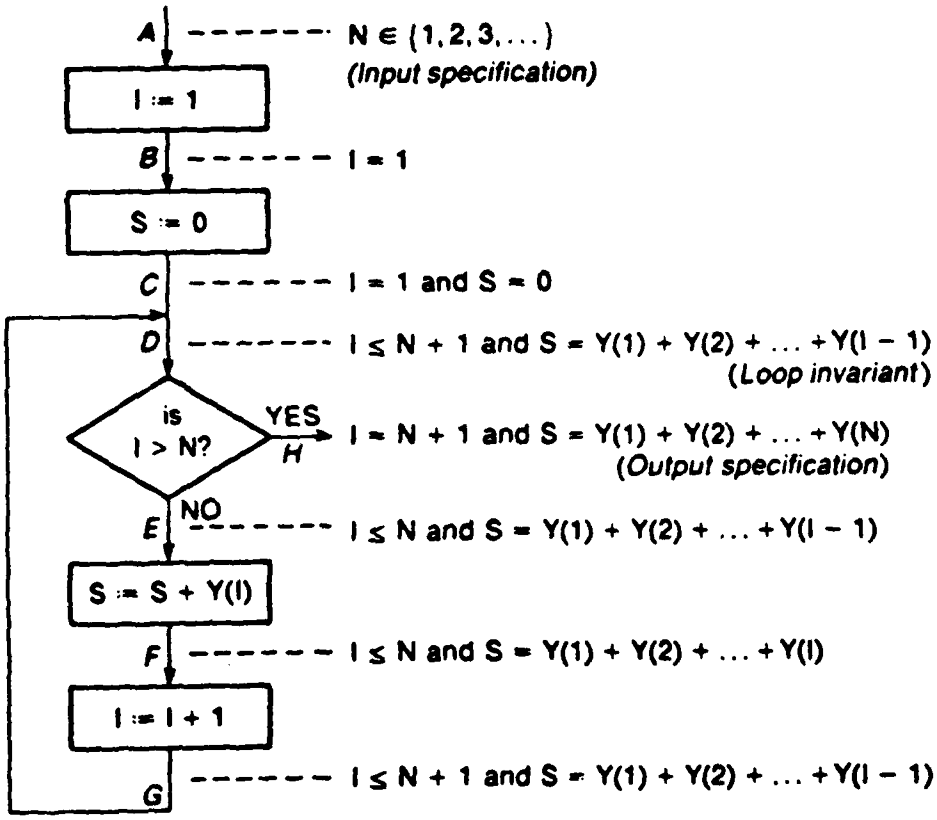


Figure 2: Example of a Proof of Correctness

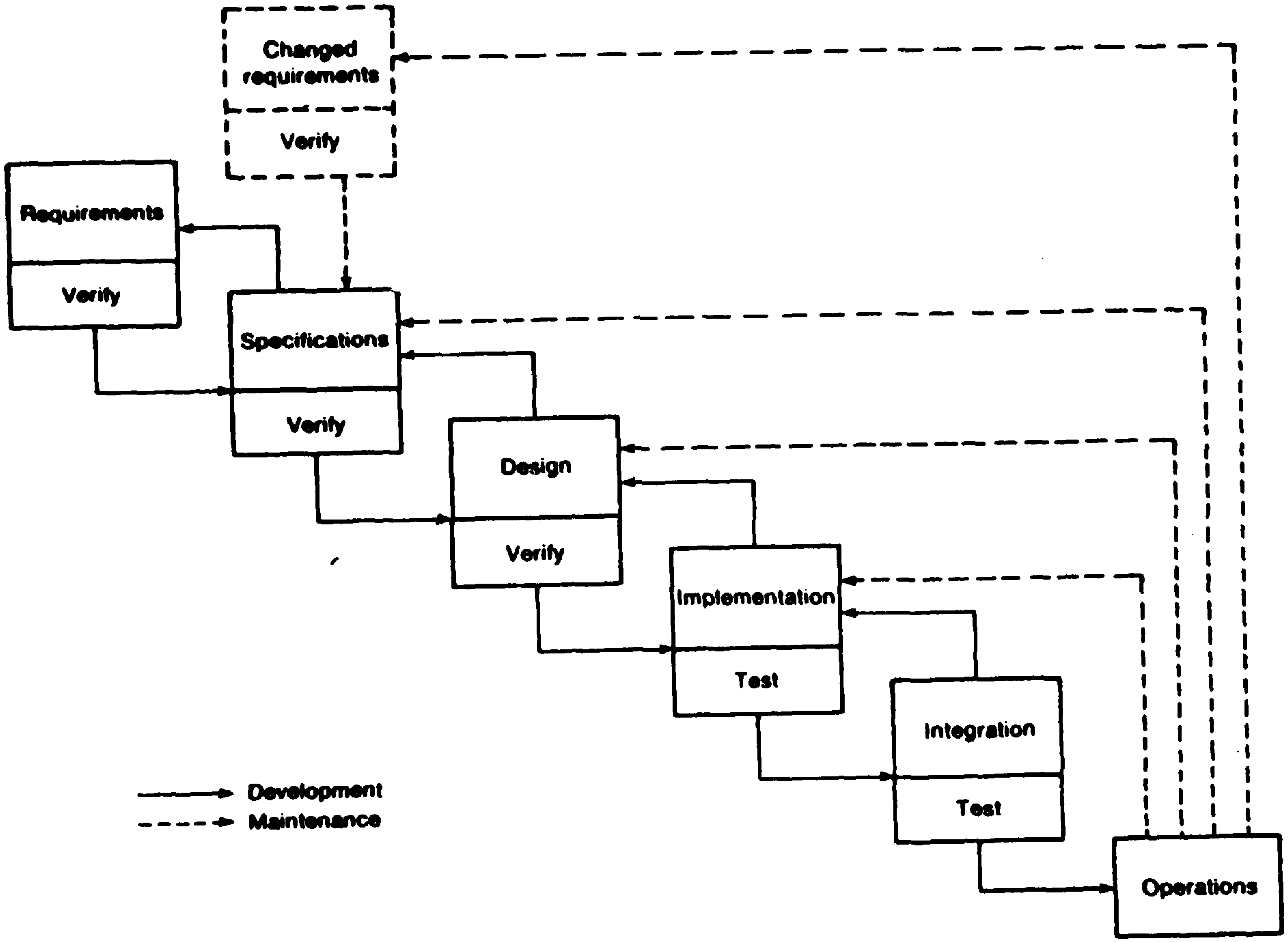


Figure 3: Waterfall Model
(Adapted from Schach, 1990)

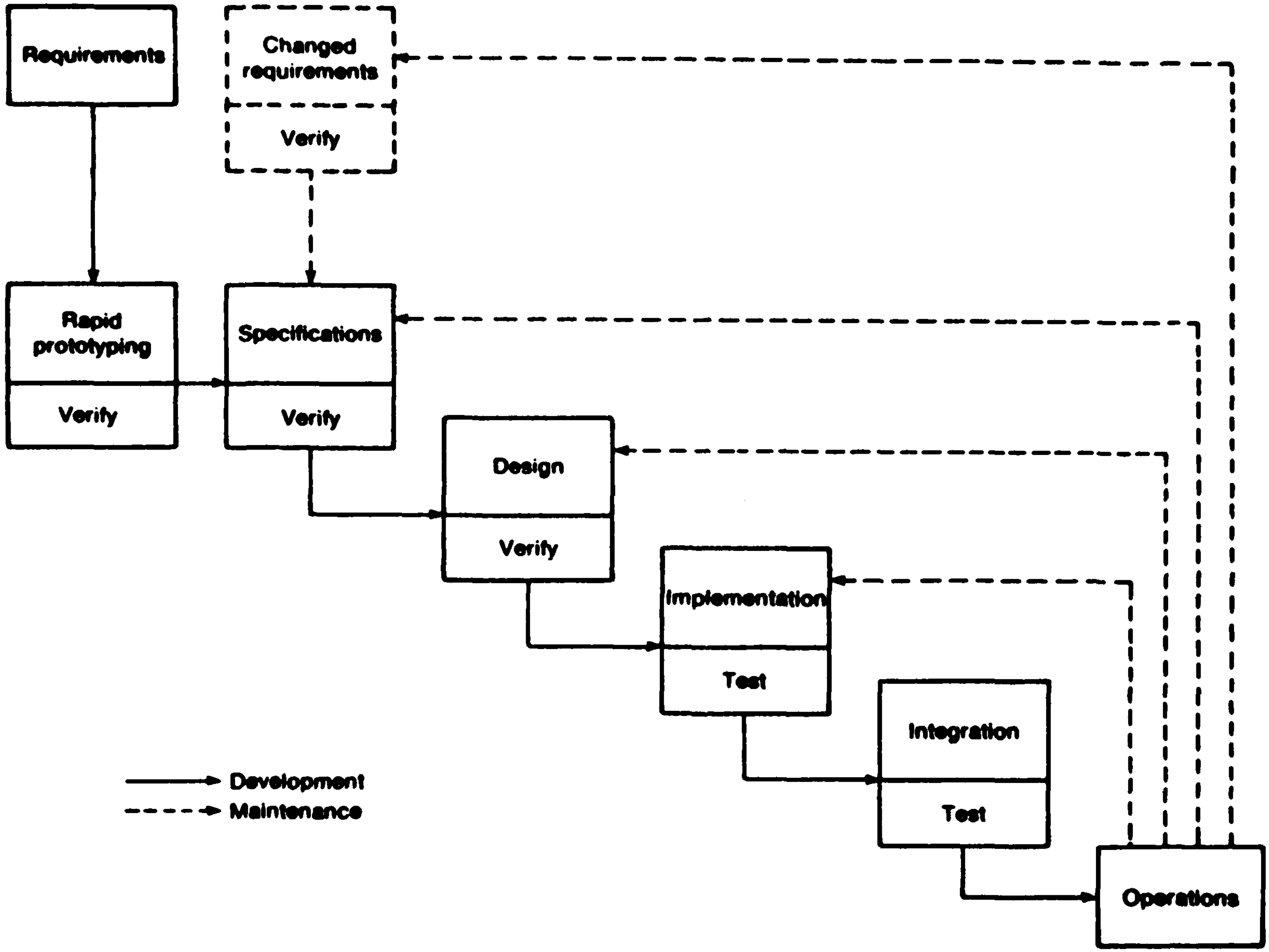


Figure 4: Rapid Prototype Model
(Adapted from Schach, 1990)

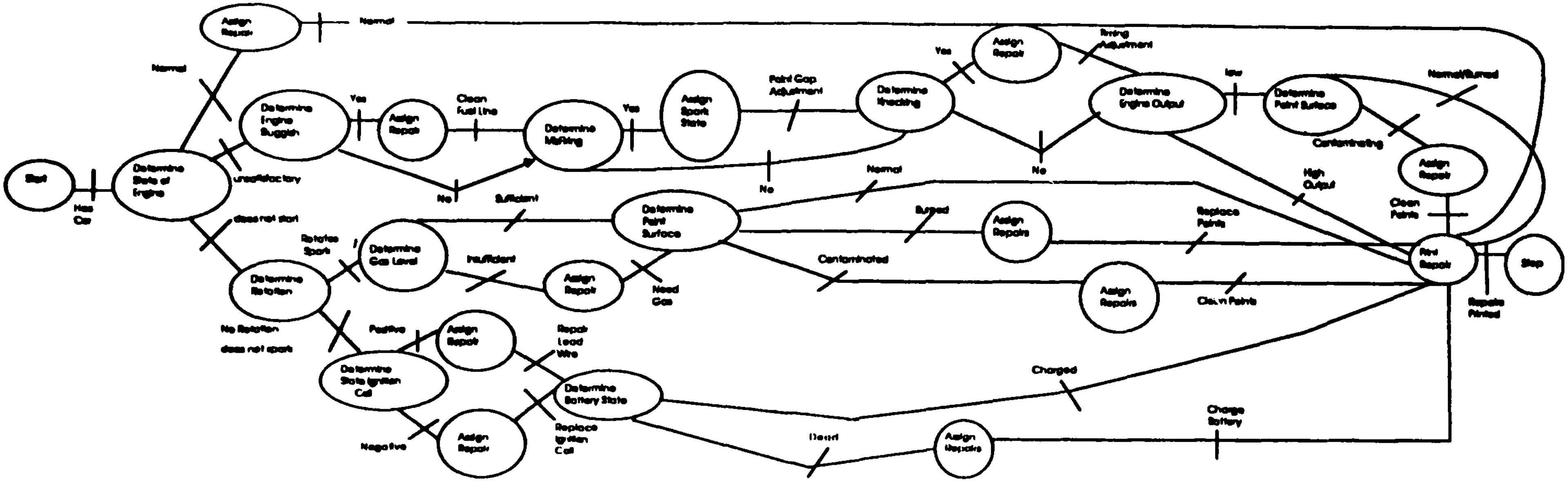


Figure 6: Example of Petri Net Representation of an Expert System

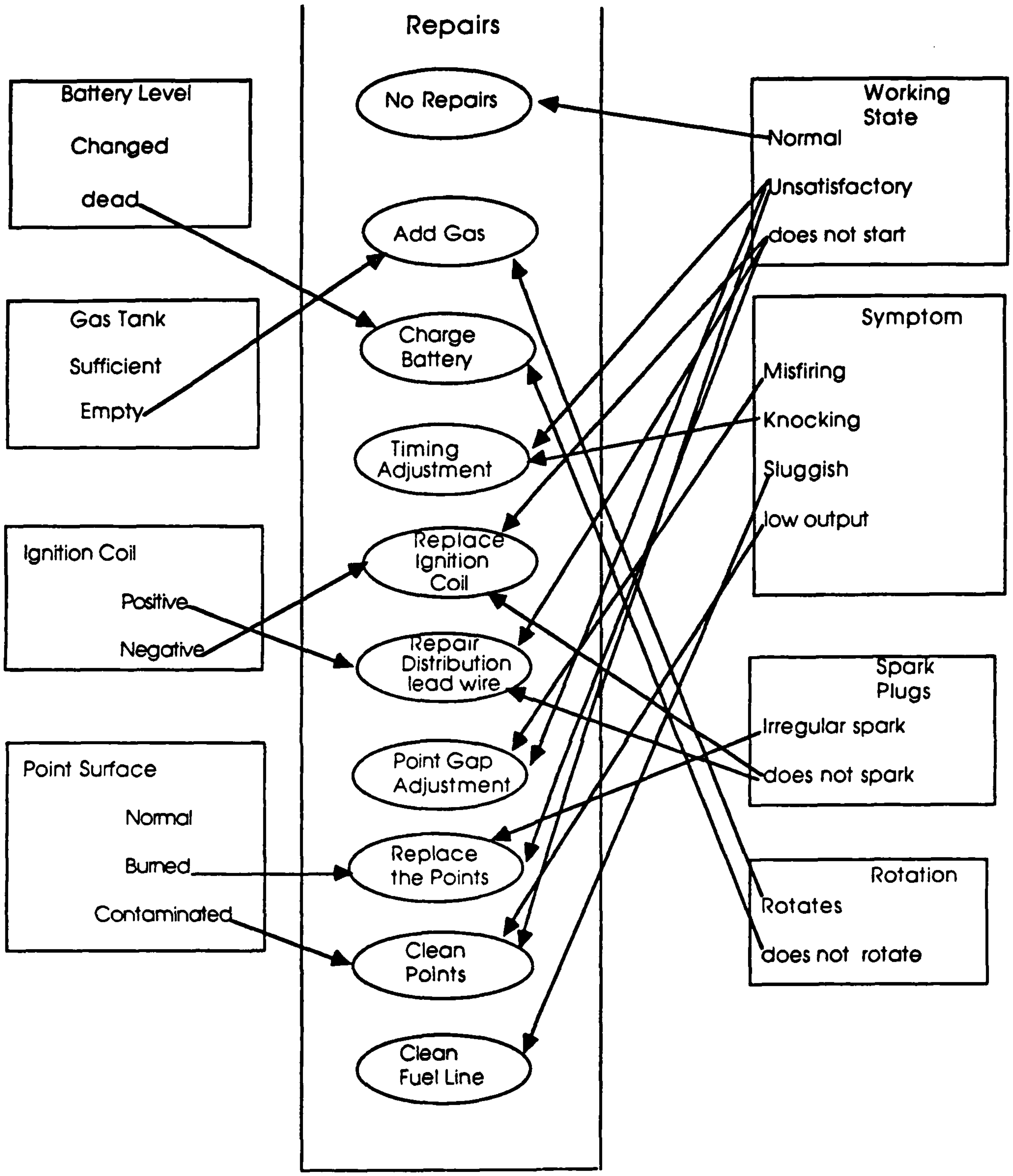


Figure 7: Example of ER Diagram of an Expert System

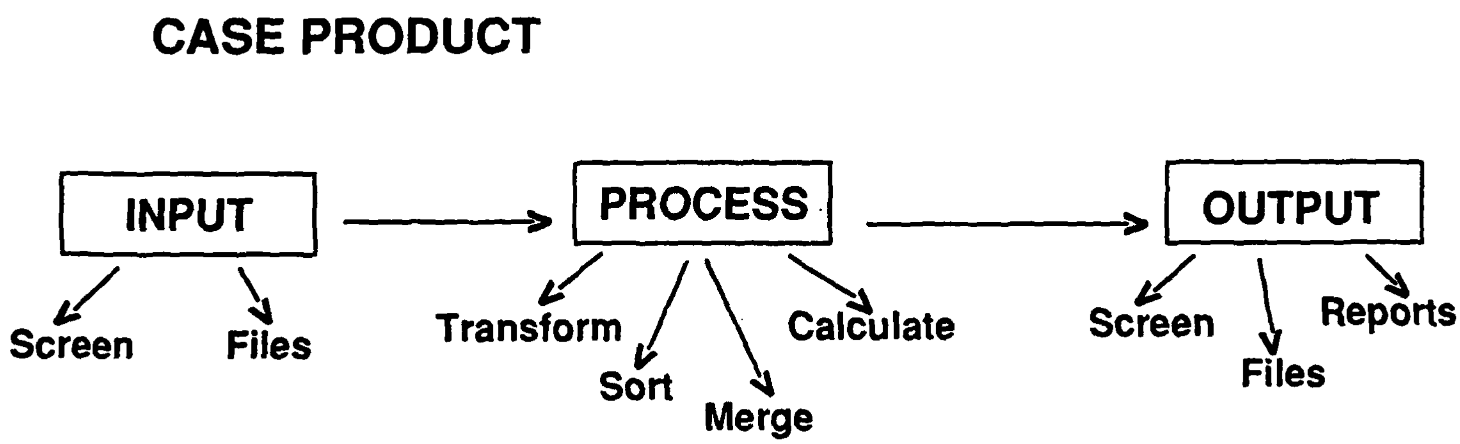


Figure 8: Objects for CASE Product

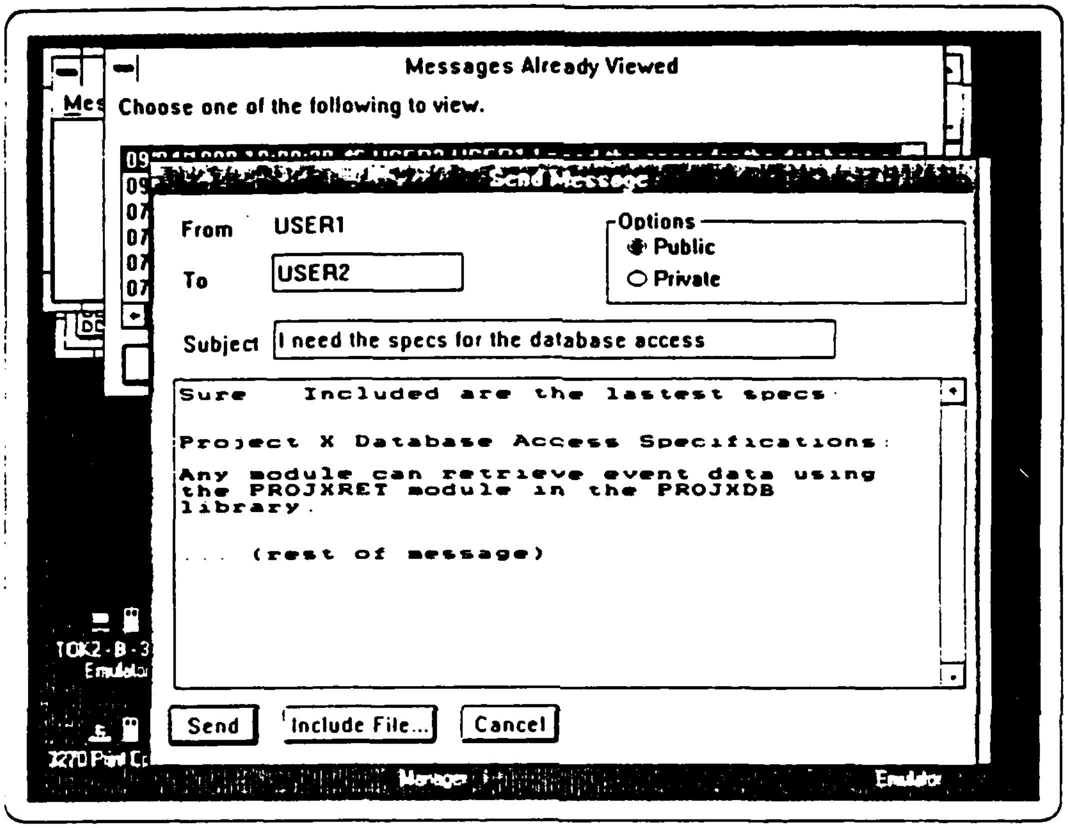


Figure 9: Screen for Computer-Supported Cooperative Environment

Assessment of Enabling Technologies for Computer-Aided Concurrent Engineering (CACE)

Azad M. Madni[1] and Amos Freedy, Perceptronics, Incorporated, Woodland Hills, California

SUMMARY

In today's competitive industrial environment, a major priority is the development of new and novel approaches for dramatically reducing product development times while improving overall product quality. The concept that has become pivotal to achieving these objectives is concurrent engineering (CE). CE offers several advantages over traditional sequential engineering including shorter product development times, superior product quality, dramatically higher acceptance, lower cost, and higher assurance of meeting time-to-market requirement. CE calls for early and regular collaboration among engineering, manufacturing, management and support personnel during the product planning and design processes.

This paper provides an assessment of enabling technologies underlying CE. It presents a conceptual framework that provides the basis for discussing the technological components. These include: a collaborative design environment, "executable" process models and design specifications, a formal approach to human-machine integration, interactive multi-media technologies and computer-aided concurrent engineering tools and their integration.

[1]Requests for reprints should be sent to Azad M. Madni, Perceptronics, Inc., 21135 Erwin Street, Woodland Hills, CA 91367.

Running Title: Concurrent Engineering Technology Assessment

INTRODUCTION

In the last three decades computer-aided automation has seen a steady increase in the different phases and facets of a product's life. The factory of the future seemed imminent. But after huge expenditures and frequent speculation, it became apparent that the design automation concepts and sophisticated equipment could not keep up with an environment plagued by constant change. Variations in raw material and equipment breakdown are but a few examples of how "change" is the rule, not the exception in manufacturing environments. Moreover, despite the manifest advantages of "soft" automation, there are still excessive delays in time-to-market that, in part, negate any advantages.

The fundamental problem is that the traditional approach to solving large complex engineering problems is inherently sequential, i.e., the problem is decomposed into its constituent subproblems and the subproblems are tackled sequentially -- moving from R&D of materials and processes, to product design, manufacture, installation, and support. This approach has obvious drawbacks including: late discovery of problems; long, costly design iterations; suboptimal solutions due to insufficient evaluation of options early in design; long product development times; concomitant negative impacts of product cost, quality, supportability; and last minute engineering changes due to design shortcomings discovered at manufacturing time.

In light of these manifest deficiencies with sequential engineering, the industry is turning to *concurrent engineering* . Winner et al (1988) offered the following definition of concurrent engineering in IDA Report R-338.

> "Concurrent Engineering is a systematic approach to the integrated concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers from the outset, to consider all elements of

the product life cycle from conception through disposal, including quality, cost, schedule, and user requirements."

This paper presents a conceptual framework for CE, along with an assessment of enabling technologies and tools that can promote and naturally enforce CE principles and practices.

COMPUTER-AIDED CONCURRENT ENGINEERING

In a concurrent engineering environment, multi-disciplinary teams consisting of different members from the design, manufacturing and support functions work together with the customer on all phases of product development, fully sharing information and participating in decision making.

The challenge of concurrent engineering is to overcome organizational fragmentation and manage complexity through a combination of cultural changes and technological innovation. CE requires a new approach to accountability, focus, and coordination of multiple objectives-oriented teams. Specifically, the introduction of CE requires that both vision and knowledge must be aligned across the different functional groups (materials, engineering, manufacturing, management, and support services).

The concurrent engineering team usually works under a single budget. Since the team has a common goal, design changes that provide an overall benefit to the team can be identified quickly. Resolution of design deficiencies does not have to be deferred, but addressed as soon as they are discovered. This strategy is key to producing a robust system that incorporates the best or acceptable features for all.

With respect to the insertion of CE methods and tools within design and manufacturing environments the challenges are managing application complexity while producing a useful product; managing cultural change while introducing CE principles and practices; managing technical and technological difficulty without losing focus; leveraging existing tools and inplace procedures, to the extent possible, without violating CE principles; and measuring incremental progress to ascertain accomplishment of interim milestones without interfering with ongoing work.

While concurrent engineering has been accepted in concept by the engineering and manufacturing communities, a unified framework for introducing CE practices, procedures and tools is necessary to feel the full impact of CE. The concurrent engineering process (Figure 1) is conceptualized from this viewpoint. The main idea behind this process is to allow the designers to look "down the line" while still in the early stages of the design process. The figure shows the overall process and sequence of design steps undertaken by the collaborative design teams in the concurrent design of a product.

Insert Figure 1 about here

Figure 1 shows a conceptual framework for CE. Starting with a preliminary set of requirements the design teams collaborate in the design process within a design environment that supports electronic mail, teleconferencing, and sharing of all "design objects" (e.g., partial solutions, design versions and tools). The *preliminary design* process is facilitated by "rough cut" process modeling and analysis tools: The design teams construct high-level process models in the application domain with the help of the process modeling tools. The preliminary design is progressively *refined* into detailed process models for indepth analysis. When the process and product design specification becomes relatively "stable", the manufacturing process is simulated and evaluated prior to making a commitment to hardware. The *process simulation* requires "executable" process models for simulating the different flows in a manufacturing enterprise. Several design versions are created during the course of this simulate-and-evaluate cycle. These are catalogued in the order created along with attendant assumptions, decisions and constraints thereby creating a design history and design version audit trail for future use by the design teams.

ENABLING TECHNOLOGIES

Realizing the full benefits of concurrent engineering is far more than a technological problem. In fact, to realize the full impact of CE requires a fundamental cultural change at all levels in an organization. The technology assessment done in this paper is from two perspectives: (1) how technology can enhance the simultaneous development of the product and the process; (2) how technology can provide the basis for realizing a much-needed cultural change. Within the overall framework of Figure 1 we discuss and assess five different enabling technologies: (1) collaborative design environments; (2) "executable" process models and design specifications; (3) human-machine integration methodologies; (4) interactive multi-media technologies; and (5) CACE tools and tool integration techniques.

Computer-Aided Collaborative Design

A key element of the design process is to ensure that all key decision-makers and system operators can participate on design teams. Since it is not usually practical to have collaborative sessions involving all technical disciplines (e.g., manufacturing operations manager, design engineers, production engineers, reliability/maintainability engineers, systems designer, and shop floor managers), the project leader or task force needs to partition tasks to be carried out by teams of 4-6 people and to plan communication and coordination needed between the design teams. To expand communication, each member of the design team needs access to and protocols for the use of terminals and servers that provide full, high resolution interactive graphics capabilities. The different terminals, workstations or mainframes need to be interconnected via an Ethernet network using the X-Window System™ standard. The members of any design team need to be supported by an environment that helps them communicate easily with each other, to coordinate their activities, and to share common design objects (e.g., design representations, design history, and design tools).

Figure 2 shows a high-level view of the Collaborative Design Environment (CDE). A large screen display facilitates focus on the issues being discussed whenever a subgroup meets face-to-face in the same room. In those situations where subgroup participants in a design session are remote from each other, inherent delays make synchronous interaction more difficult, so that more structured protocols are necessary to guide orderly discussion and decision-making. The open systems that have become available recently involving heterogeneous servers and workstations using UNIX™, Ethernet, and X-Windows have made possible several interactive functions that were previously infeasible. The process can be helped by teleconferencing technology, including FAX and video hookups. Of crucial importance to this work is the extent to which each participant can be made aware of changes to design objects and the ease with which each can access the state of such objects in a global data base. Figure 3 shows the functionalities required from the collaborative design environment.

Insert Figure 2 about here

Insert Figure 3 about here

While collaborative design is a well-received concept, there are still some technical hurdles (e.g., the ability to share design "objects") that have to be overcome to develop an effective Collaborative Design Environment (Mujica et al., 1990).

Executable Process Models

Process models allow description, integration, and evaluation of the different "flows" in a manufacturing enterprise at different levels of detail and from multiple perspectives (Madni et al,, 1990; Estrin et al., 1986).

Process models, or models that produce executable specifications (Madni et al., 1990; Harel et al., 1988), allow the design team to analyze the impact of "downstream" constraints on candidate designs with a view to achieving a design that satisfies manufacturing, assembly, cost and support constraints. Table 1 provides a summary of the desired characteristics of process modeling and simulation.

Insert Table 1 about here

Human-Machine Integration

In the foreseeable future, humans will continue to serve as "enabling components" in a manufacturing enterprise. Proper integration of humans and equipment can make all the difference between a relatively trouble-free factory and one that is beleaguered with human errors arising from a poor integration of humans and machines. Madni's (1990) approach to human-machine integration relies on four different classes of simulation. This family of simulations is designed to produce the most cost-effective solutions at various stages of the concept development/exploration and demonstration/validation phases of the design process. Table 2 provides a comparison of the applicable simulation approaches, their respective strengths/advantages and cost impacts. As shown in this table, a staged simulation methodology provides the basis for an effective human-machine integration approach. Each of these simulation stages are discussed below.

Insert Table 2 about here

Analytic Simulation, the first level of simulation, is directed to modeling all "flows" in a manufacturing process with heuristic/rule-based models of human operators working with simulated machine counterparts (Madni, 1988b). From a human-machine integration perspective the purpose of this simulation is to determine operator workload with different function allocation options and levels of automation.

Designer/User-in-the-Loop Simulation is the second level of simulation that employs a real operator (versus a model) working with the simulation. The purpose of this level is to solve man-machine integration problems, uncover deficiencies in the overall concept of operation, and to refine the human behavior models within the analytic simulation. This level pertains to the "horizontal prototyping" phase in interactive system design (Madni, 1988b).

Designer/User-in-the-Loop Prototyping is the third level of simulation. The purpose of this level is to demonstrate selected functionality of the overall system for designer / operator review and feedback prior to the systems integration task.

Users-in-the-Loop Networked Simulation is the level associated with the command and control of an automated factory. From a man-machine integration perspective, this level of simulation analyzes individual operator's communication and coordination load and error patterns while operator perform their assigned tasks.

Interactive Multi-Media Technology

Interactive multi-media technology has opened up a whole new dimension in human-machine relations and human-human collaboration. Specifically, advances in multi-media storage and delivery have expanded the range of options available for teleconferencing, collaborative design, embedded training and education. Figure 4 summarizes a few key capabilities and sample uses of the different components of a multi-media environment. Today, it is widely believed that IMT will realize the much needed shift in paradigm in systems design and manufacturing. For example, by merely incorporating a "live video" window in a design workstation so that each individual designer can see and hear another (in a designated window) as opposed to interact through text messages can contribute to teamwork and bring about this much-needed cultural change. Similarly, 3-D animation video such as DVI, CD-I can promote visualization in educational workstations. Table 3 provides examples of some effective uses of the different multi-media technologies.

Insert Figure 4 about here

Insert Table 3 about here

Computer-aided Concurrent Engineering Tools and Their Integration

As designs evolve, the design team needs appropriate tools to support the different design activities. A broad array of analytic, heuristic, empirical and simulation-based tools are required to achieve the collective objectives of a product design within a CE rubric. CACE tools are a set of software tools on graphics workstations that help the collaborative design team visualize and analyze how their design will be built, tested and introduced on the shop floor. Computer-aided concurrent engineering (CACE) tools are the "productivity multipliers" in CE. CACE tools serve various purposes and different users. For the tools to be effective they must be delivered on the host environment-compatible platform and language. Figure 5 provides an overview of the major elements of CACE.

Insert Figure 5 about here

The *System Design and Modeling Toolkit* subsumes all structural models (e.g., assembly design, task modeling, data base generators), behavioral models (e.g., manufacturing process simulation, performance analysis) and tradeoff analyses models (e.g., cost-benefit analysis, yield-to-performance analysis). This toolkit includes both process modeling and product modeling tools.

The *User Interface Generation Toolkit* comprises prototyping, programming and "storyboarding" aids. Specifically, visual programming tools, windowing, screen layout and dialogue design tools fall under this category.

The *Human-Machine Integration Tools* consist of human-machine function allocation tools, task-imposed workload analysis tools and user interface analysis tools (e.g., complexity analysis, scripting and animation, simulation, and visualization).

The Systems Integration and Extensibility Tools consist of horizontal integration and vertical integration tools, and various combinations of the two (Madni, 1988b).

Despite the existence of numerous tools directed at facilitating the development process, a major impediment to realizing a comprehensive solution is the current state of tool integration. CE software tools may be referred to as integrated simply because they share a common user interface or because a vendor offers tools that support more than one phase of the software development life cycle. Some speak of tool integration in terms of the support of shared data storage (Wasserman, 1988), while others describe a fully supported development life cycle (Martin, 1990) where all tools interface to a common framework/database in a distributed environment (Phillips, 1989).

Tools cannot yet be defined as fully integrated, i.e., both control integration and data integration. Most tools / toolsets could be defined as data integrated (or, more appropriately "joined") with an underlying object database. The tools may even share this "dictionary," but may be limited by any level of scope, interoperation, or environmental capabilities. These tools may not represent or interact with the multiple views or phases of objects required in coordinated, heterogeneous data storage. Most tools also do not incorporate the external control interfaces and formats that are required for automatic inter-tool process / data flow.

There are different views on what constitutes tool integration and at what level tool integration should be considered for a specific application or project. Specifically, there are five classes of tool integration:

- Internal integration, i.e., integration of tools and data of a single vendor.
- External integration, i.e., integration of tools from multiple vendors.
- Environment integration, i.e., integration of tools with the operating environment.
- Process integration, i.e., integration of the development process-related activities.
- End-user integration, i.e., integration of tools with their end user group.

Internal Integration. Internal integration is the integration of tools and data of a single vendor. Such tools use either a local data dictionary or are "joined" through a central, shared dictionary. The data dictionary is invariably a type of (relational) database that allows a vendor to offer consistent data for the tools. The resultant product is generally proprietary.

The primary problem with internal integration of single-vendor tools is the limited range of tools offered by the particular vendor. Vendors generally do not offer a full complement of (integrated) tools. Further, the tools tend to be targeted to either personal computers or workstation / mainframe environments. In those rare instances when a single vendor offers a full range of tools, the user may not find the different tools equally useful because of some inherent limitations. Tool users find it unacceptable to be restricted to a single-vendor tool suite due, in part, to these limitations.

The single-vendor aspect also impacts the level of flexibility in a toolset from which a user can benefit. If a vendor allows a user to modify the interfaces/objects of such a nonstandard toolset, it could very well further widen the "compatibility gap" with other (vendor) tools. Even if conversion capabilities are developed to allow a database/interface to be adapted to a standard format, the ultimate responsibility for adapting the modifications may well be left to the user.

There are also problems inherent in the use of relational databases as the basis of a software tool data dictionary (Brown, 1989; Chappell et al., 1989). Relational databases cannot readily accommodate the flexible complex object/data types (e.g., code segments, design diagrams, user processes) required by CACE technology. They are also generally unable to handle the amount of data that it takes to implement a fine-grained object management system (i.e., objects composed of data items and interrelationships that are more complex than the level afforded by a singular file or character string representation). The problem here is one of data size and performance characteristics. The net result is that: (a) users are left with a single, limited view of the data objects and (b) users find it difficult to share data objects among the tools.

This recognition has spurred many vendors to publish their tool interfaces as a means of promoting greater acceptance from the tool user community (Gibson, 1989; Wasserman, 1988). While this helps users (and other vendors) interface their own tools with the tools of a specific vendor, it fails to address the larger problem of data incompatibility among tools.

Efforts are currently underway to expand the integration of object-oriented databases (OODB) with tools. An OODB is the basis of next-generation "repository" products currently under development by both IBM and DEC. OODB differ from relational databases in that they store data maintenance and access rules along with the object data. This technology would provide extended capabilities (e.g., multiple object views, modifiable rules, and object types) and would enhance the distribution/performance aspects of a shared data dictionary (or repository). Universal acceptance of OODB technology is impeded by the fact that the technology is relatively new (no single data model) and by the prior commitment to relational databases of many vendors, users, and standards organizations.

External Integration. "External" integration pertains to the integration of tools from one vendor or with those of another vendor. This integration is usually in the form of "access control" and/or "data control." With "access control," the tools of one vendor can be invoked by tools from other vendors. The tool invoked returns appropriate messages/codes to the invoking process/tool. With "data control," external tools are allowed to indirectly manipulate the data contained in the internal dictionary of a tool.

Due to the development of proprietary interfaces, most vendors cannot readily interface with other vendor databases. Not only might the format of the data be different, but the contents of the dictionaries may be inconsistent. The weaknesses of relational databases are also exacerbated with objects are transformed. Since data rules are imposed by the tools (as opposed to the database), the consistency of objects (the "view") may well be distorted as semantic content is lost/misinterpreted when moving data between tools.

Another problem with external integration directly underlies the importance of the selection and acceptance of standards. The integration of the tool/data interfaces from two different vendors is a costly proposition that includes both the indirect support and maintenance associated with the tools and interfaces of both vendors. Ultimately, given the current state of standardization efforts, the end users themselves may have to act as systems integrators and write the code necessary to effectively integrate a tool from different vendors. This assumes that tool interfaces are well documented and that the user can afford the integration of time/cost.

Work is currently underway by IBM, DEC, and others to define a central data repository that would take steps to solve the data incompatibility issue (3, 27). A repository is a common shared database that stores the rules (or relationships) associated with tool data objects. The data may either reside in the repository or be distributed throughout an application network.

Environmental Integration. Basically, tools are tied to their environment through the host operating system or hardware platform. The tools may use specific features of the operating system or support toolset that are unavailable on other systems (e.g., windowing system variants, Unix system variants). This makes it harder (and in some cases impossible) for these tools to be rehosted to other environments.

There are also the problems associated with scalability when considering a move from one environment to another. Some tools might experience performance degradation or bump up against environment constraints (e.g., memory requirements, CPU characteristics, data storage facilities). Tools developed for a single-user PC environment may be unable to support multi-user projects because of concurrency or project/data size requirements. For example, it may be impossible to run multiple instances of the tool or the tool may be unable to handle the increased data access requests.

Another environment-related issue is that of tool integration with a configuration management system. No viable tool offering (regardless of the scope of the toolset) can overlook the importance of maintaining an ordered version control history for data objects. Some tools integrate a simple versioning scheme for file objects, but the concept must be extended to include all data types as object granularity becomes finer than the file level and as source code becomes more a "derived object" as opposed to a central entity. Extension of tools to include configuration support must be given careful consideration so as not to trigger scalability problems (particularly data storage and performance limitations).

One possible solution to the environment integration issue is the emergence of a standard for the Unix operating system. Unix has been showing promise as a "cross-over" operating system catering to the needs of both the technical and commercial markets (Cortese, 1990; Cureton, 1988). In this respect, tool vendors would be relieved of the burden of maintaining separate product lines for multiple operating systems by targeting the Unix system. This would also alleviate many of the rehosting issues facing the vendors as product porting would become solely a hardware (Unix system platform) issue. Environment integration and tool/database distribution would be greatly enhanced through the accessibility of existing network support functions (e.g., NFS, TCP/IP).

Process Integration. In anticipation of smoothing life cycle process transients, software tools are being developed to help combine the various phases of the entire life-cycle (e.g., project management, analysis and design, configuration management).

The Integrated Project Support Environment (IPSE) offerings allow for tool data, control, and presentation integration (Figure 6). Data integration refers to the coordination of access to the underlying tool database(s). Control integration refers to the coordination of access to the tools themselves. Presentation integration refers to the coordination of the user interface. The basis of data integration is the repository. The basis of control integration is the software "backplane" or executive that provides the requisite interfaces to the different tools. IPSE allows for mixed operating system support and, in some cases, for mixed platform support.

Insert Figure 6 about here

A key issue in process integration is that of tool flexibility. Users demand that tools be easily modifiable and customizable to their particular needs. Most tools remain closed to such customization. Those that claim an "open" interface generally allow modification of merely the presentation characteristics. For tools to be fully integrated, users must be able to modify the characteristic behavior of the tool (e.g., design rules, object rules, object types, process), not just the user interface (Forte, 1989). Users should not have to abide by a strictly imposed process for software development (e.g., waterfall model, spiral model) if they are best served by some internally developed or hybrid process.

Another key problem is that most tools support only very specific design methodologies. Also, the methodology supported by the tools is usually strictly enforced. Consequently, the users have to select and learn a new methodology or choose from a limited set of tools that support their current methodology. In the interest of accommodating in-place methods and procedures, tools need to be able to expand beyond the traditional "bottom up" or "top down" design approaches and allow configuration modifications to support alternative ("middle out") methods.

Another term used in the tool integration arena is "framework." Frameworks are essentially tool backplanes with a tool management executive. The executive handles the overhead involved with coordination of the tool suite (e.g., user presentation, tool registration and instantiation, error reporting). At the bottom end of the framework is a common data interface/repository, messaging system, and operating system services manager. These interfaces unburden the tool modules of the particulars of the host environment and allow for a broader, more interchangeable product set.

User Integration. The concept of integration with the end user ranges from something as simple as maintaining a consistent user interface to something as complex as providing support for an expert system interface. In particular, a standardization (also known as presentation integration) could help enhance user acceptance of tools.

One of the basic concerns of tool integration tools in general, is the issue of a consistent user interface/presentation (Forte, 1989; Phillips, 1989). The learning curve associated with adopting a new tool is steep and requires a significant investment on the part of an organization that goes well beyond the cost of the tools. It involves the time and cost of comprehensive training and support (both from the vendor and from the users). Standardized user interfaces and tool function could help keep these costs under control as well as offer greater tool/choice flexibility to the users.

In addition to the underlying system, user interface consistency is also becoming a key issue to tool vendors. The user interface contributes significantly to the acceptance and learning time for a product. It would seem that the user interface is the easiest of the integration areas on which to standardize so vendors must use caution not to promote a "quick and dirty" solution. If the vendors do not carefully analyze the requirements of the user interface without taking into account the context of the tool usage, they may end up with an interface standard that promotes consistency at the expense of ease of use.

Figure 7 shows a full IPSE model that "pulls together" all the components of a tool environment in a single framework architecture.

Insert Figure 7 about here

Technology Introduction Strategy

As with any new technology, the introduction of concurrent engineering methods and, more specifically, computer-supported collaborative work (CSCW) can be expected to face some degree of resistance from users despite the potential benefits.

Gould and Lewis (1985) recommend that "early in the development process, intended users should actually use simulations and prototypes to carry out real work, and their performance and reactions should be observed, recorded, and analyzed" (p.300). This is all the more significant for CSCW because of the added task complexity. When the development cycle involves individual users performing single tasks (e.g., using a spreadsheet application), design errors emerge relatively quickly because the interactions are limited to one person and one system. However, when the system supports the cooperative work of multiple users, a higher level of complexity is involved. Interactions among multiple users create a set of inter-dependencies not found in single-user systems. As a result, design errors emerge more slowly and are more difficult to pinpoint. Additionally, there is a greater opportunity for unintended effects, some of which may not appear for a long time! We plan to exploit this performance-based analysis approach in facilitating technology transition.

Also, human factors will play a significant role in ensuring the acceptance of CSCW. For small scale implementation, the guidelines for implementation, and products/training courses in support of the implementation process, will be instituted. For large-scale implementation, we may provide on-site consulting which may prove to be cost-effective. Geirland (1986), recommends including seven strategies in large scale implementation efforts (Table 4).

Insert Table 4 about here

Antonelli (1988) suggests that design decisions are more sound when based on input from real users. In this regard, the identification of cooperative work modes is useful for developing CSCW/CACE tools. We intend to adapt the taxonomy of cooperative work styles (Johnson-Lenz and Johnson-Lenz, 1982) for teleconferencing for our program. The key elements of the taxonomy are summarized in Table 5.

Insert Table 5 about here

In light of the foregoing considerations, we have identified the key elements of a successful approach to technology introduction in IRFPA manufacturing environments that is grounded in the key concepts summarized in Table 6. Each of these key concepts are discussed in the following paragraphs.

Insert Table 6 about here

<u>Start Small, then Expand</u>

Taking on the total manufacturing environment as the target environment is much too big a task. Our approach is to identify high payoff targets of opportunity for the insertion of CE methods and tools. This approach not only makes the problem tractable but also gives some evidence of where to set our sights next. Successful insertion of CE within certain key processes while exploiting the natural parallelism that exists in various tasks will allow us to generalize the lessons learned to some extent to include other assembly and integration function. While we intend to "start small," we do so against the backdrop of the "big picture" to assure relevance and design impact of our products.

Staged Series of Demonstrations

A key theme of this effort will be staging a series of demonstrations against the backdrop of the "big picture" to showcase technologies, applications, and evaluate work in progress. The demonstrations will include: proof-of-principle demos (e.g., process models), concept of operation demos (e.g., computer-supported collaborative work), tool usage demos (e.g., data base generators, and LAN generators), "interoperable" tools demos (e.g., assembly, and support tools with integration support tool), process visualization demos (e.g., graphical user interfaces), and simulations of integrated factory subprocesses (e.g., DEWAR assembly).

Indoctrination of End Users

Given the heterogeneous nature of the design team, and the fact that introduction of concurrent engineering is not just a change of approach but a change of culture, it is imperative that the "end users" be provided with the "big picture" along with the specific objectives and their respective roles in the design process. Without a shared conceptual model of the end objectives, and preparation for change, the process of overcoming user resistance can be truly formidable. At the level of individual tools, it is equally important to show end users how the tool fits in or is different from the status quo.

Storyboarding

The objective of the storyboarding phase is to develop a set of sequential static interface screen layouts corresponding to the preliminary concept of operation of a tool, a device or a subsystem (Madni, 1988). These series of screens with appropriate textual and pictoral annotations serve as the basis for communicating the systems' functionability to potential users. Specifically, the screens serve to communicate to the user what the tool/system does versus what the user is expected to do. Storyboarding provides the first indication of the level to which the tool/system can be expected to aid, train, or off-load the user. In addition, the initial screen compositions become a point of departure for soliciting user comments on improving the utility and composition of the screen layouts -- the user identifies missing information and/or all information elements that are best presented as exceptions instead of rules. At this stage users can indicate the specific types of help and/or software alerts they prefer as they "walk through" the tool/system storyboards. In sum, storyboarding serves as an effective means for specifying interactive software operation. Prototyping efforts can start once all pertinent user comments are incorporated into the storyboards.

Horizontal Prototyping

This particular prototyping strategy pertains to the high-fidelity replication of the user interface of the final system with simulated functionality and times (Madni, 1988). The purpose of horizontal prototyping is to provide a vehicle for identifying shortcomings and errors in user-system interactions. Insofar as CE tools are concerned, these deficiencies are in the form of missing or extraneous information, inordinate time delays in user-system interactions, suboptimal windowing and screen layouts, and so on. Horizontal prototyping greatly enhances the tool's overall usage concept and supports early demonstrations of the evolving functionality of the tool.

Vertical Prototyping

Vertical prototyping is the high-fidelity implementation of selected functions for user examination and feedback. The purpose of vertical prototyping is to define and implement in detail those functions are are important to overall system operations and for which user inputs are critical to improving system implementation (e.g., assumptions, algorithms). Horizontal and vertical prototyping can often be done concurrently with the results presented in the same demonstration (Madni, 1988).

Exploitation of In-place Procedures and Tools

One of the key concerns in the introduction of CE methods, practices and tools is that one does not disrupt ongoing activities and working in-place procedures. To this end, an analysis of procedures and tools in use within the manufacturing environment should be undertaken and a view to tailoring the introduction of CE, to the extent possible, to either subsume these procedures and tools or be compatible with them.

End Users Involvement in all Phases

A key concern in the introduction of new technology is user acceptance. To bias the odds in this area end users should be involved as key contributors in all phases of design. This strategy will not only increase user acceptance but, in fact, highly effective solutions may very well come from end users who have an informal database of lessons learned.

CONCLUSIONS

CE has emerged as a new way of doing business in the design and manufacture of new products. While CE has been accepted in concept, the full impact of CE can only be felt when all the enabling technologies and tools are in place. This paper discusses collaborative design environments, process modeling and simulation, human-machine integration, interactive multi-media technology, and computer-aided concurrent engineering tools and their integration -- five key technological components that must be implemented prior to successfully introducing CE approaches, practices and procedures.

Collaborative design environments are key to supporting teamwork but some technical hurdles (e.g., the problem of sharing design objects) have to be overcome. *Process modeling and simulation* potentially provides members of the design team with the ability to "look down the line" while still in the conceptual design phase. However, modeling, integrating, and displaying all the different "flows" in a manufacturing enterprise without overwhelming the users continue to be a major challenge. A systematic methodology for *Human-Machine Integration* is key to successful human-machine performance. The suggested simulation-based approach to human-machine integration is both methodical and cost-effective. Interactive multi-media technology holds great promise as a motivator and a precursor to the required cultural change. The specification of a *computer-aided concurrent engineering* (CACE) toolkit that spans the life cycle of the product will not only reduce designer time-on-task, but also provide an audit trail of design decisions and "lessons learned". Tool integration continues to be a major challenge but with the emergence of standards, this problem should become more tractable than it is today.

REFERENCES

Antonelli, D.C. (1988) Research and product usability. Human Factors Society Bulletin, 31(11), 2-4.

Brown, A.W. (1989) Database Support for Software Engineering, New York: Wiley.

Chappell, C., Downes, V., Tully, C. (1989) Real-Time CASE: the Integration Battle, Ovum.

Cortese, A. (October 1989) DEC Challenges IBM CASE Strategy, Computerworld.

Cortese, A. (January 1990) CASE Standard Comes Overseas for Unix Arena, Computerworld.

Cureton, B. (March 1988) The Future of Unix in the CASE Renaissance, IEEE Software.

Estrin, G. (1978) "A Methodology for Design of Digital Systes - Supported by SARA at the Age of One," AFIPS Conference Proceedings 47, 313-324.

Estrin, G. Fenchel, R., Razouk, R., & Vernon, M. (February 1986) "SARA (System ARchitects Apprentice): Modeling, Analysis, and Simulation Support for Design of Concurrent Systems," IEEE Transactions on Software Engineering SE-12, 293-311.

Forte, G. (1989) In Search of the Integrated CASE Environment, CIA/S/E Outlook 89, No. 2.

Geirland, J. (1986) Macroergonomics: The new wave. In _Office systems ergonomics report_, 5 (pp. 3-11). Santa Monica, CA: The Koffler Group.

Gibson, S. (March 1989) Some Win, Some Lose When Repository Debuts, _Computerworld_.

Gould, J.D., and Lewis, C. (1985) Designing for usability: Key principles and what designers think. _Human Aspects of Computing_, 28, 300-311.

Groupware: Putting teamwork in the PC. (1988) _PC Week_, 65, p. 81.

Harel, D., Lachover, H., Naamad, A. Pnueli, A., Politi, M., Sherman, R., & Shtul-Trauring, A. (April 1988) "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," _Proceedings of the 10th International Conference on Software Engineering_, Singapore.

Johnson-Lenz, P. and Johnson-Lenz, T. (1982) Groupware: The process and impacts of design choices. In E.G. Kerr and S.R. Hiltz (Eds.), _Computer-mediated communication systems_ (pp. 45-55). New York: Academic Press.

Madni, A.M. (May 1988) HUMANE: A Knowledge-Based Simulation Environment for Human-Machine Function Allocation. _Proceedings of IEEE National Aerospace & Electronics Conference_, Dayton, Ohio.

Madni, A.M. (1988) The Role of Human Factors in Expert System Design and Acceptance, _Human Factors_, 30(4), 395-414.

Madni, A.M. (August 1988) The Role of Human Factors in Expert Systems Design and Acceptance. In _Human Factors Journal_.

Madni, A.M. (August 1988b) HUMANE: A Designer's Assistant for Modeling and Evaluating Function Allocation Options. _Proceedings of Ergonomics of Advanced Manufacturing and Automated Systems Conference_, Louisville, KY.

Madni, A.M., Balcerak, R., Estrin, G., Freedy, A., and Melkanoff, M.A. (February 1990) Computer-Aided Concurrent Engineering (CACE) of Infrared Focal Plane Arrays (IRFPAs): Emerging Directions and Future Prospects. Invited paper, Second National Symposium on Concurrent Engineering, Morgantown, W.Va.

Martin, J. (January 1990) Integrated CASE Tools a Must for High-Speed Development, PC Week.

Melkanoff, M.A., and Bond, A.H. (April 1988) "Bridging the CAD-CAM Gap", Invited paper presented at the Fourth International Conference in Metal Cutting, Non-Conventional Machining and their Automation, Beijing, China.

Mujica, S. (December 1989) A Computer-based Environment for Collaborative Design. Ph.D. Dissertation Prospectus UCLA Computer Science Department.

Mujica, S., Berson, S., Estrin, G., Eterovic, Y., Leung, P., and Wu, E.H. (March1990) Architecture for Sharing Design Objects in the UCLA SARA Collaborative Design Environment, Computer Science Dept, Univ of Calif, Los Angeles.

Myers, B.A. (January 1989) "User-Interface Tools: Introduction and Survey," Software 6, 15-25.

Phillips, B. (December 1989) A CASE for Working Together, ESD: The Electronic System Design Magazine.

Wasserman, P.I. (December 1988) Integration and Standardization Drive CASE Advancements. Computer Design.

Winner, R.I., Pennell, J.P., Bertrand, H.E., and Slusarczuk, M.M.G. (December 1988) "The Role of Concurrent Engineering in Weapons System Acquisition, Institute for Defense Analyses Report, Task T-B5-602.

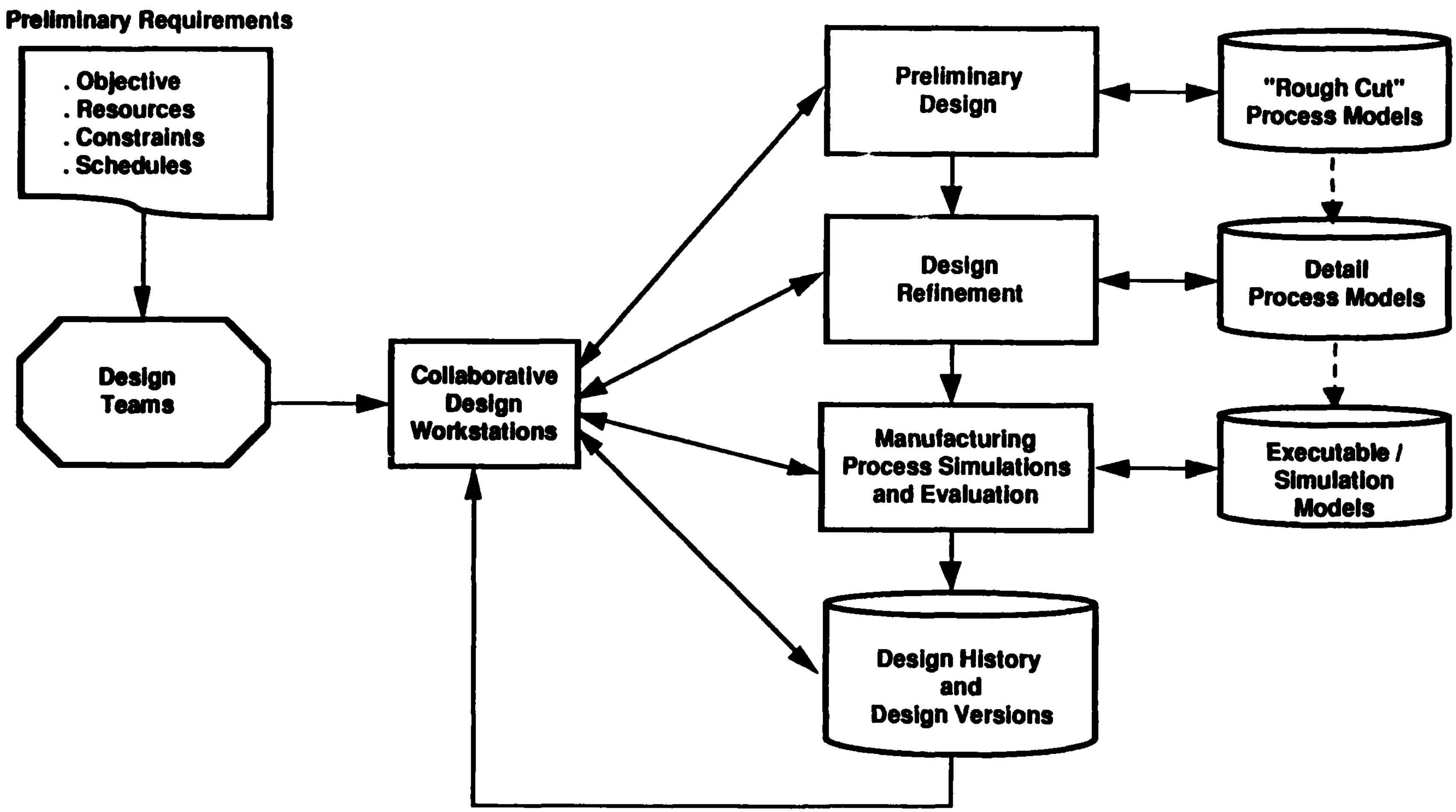


FIGURE 1 Concurrent Engineering Paradigm (Madni, 1990)

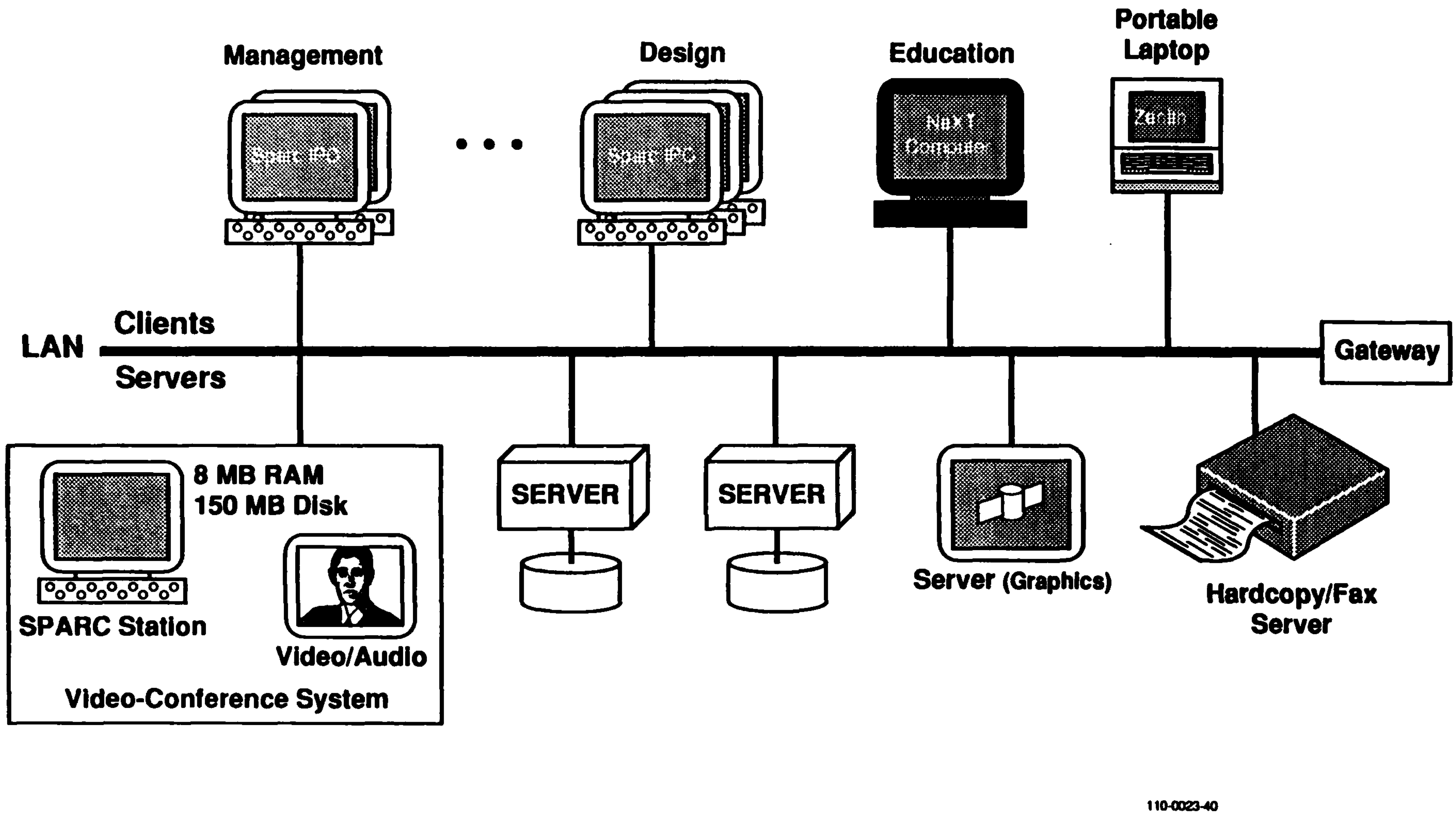


FIGURE 2 Networked Client-server Collaborative Design Environment Architecture

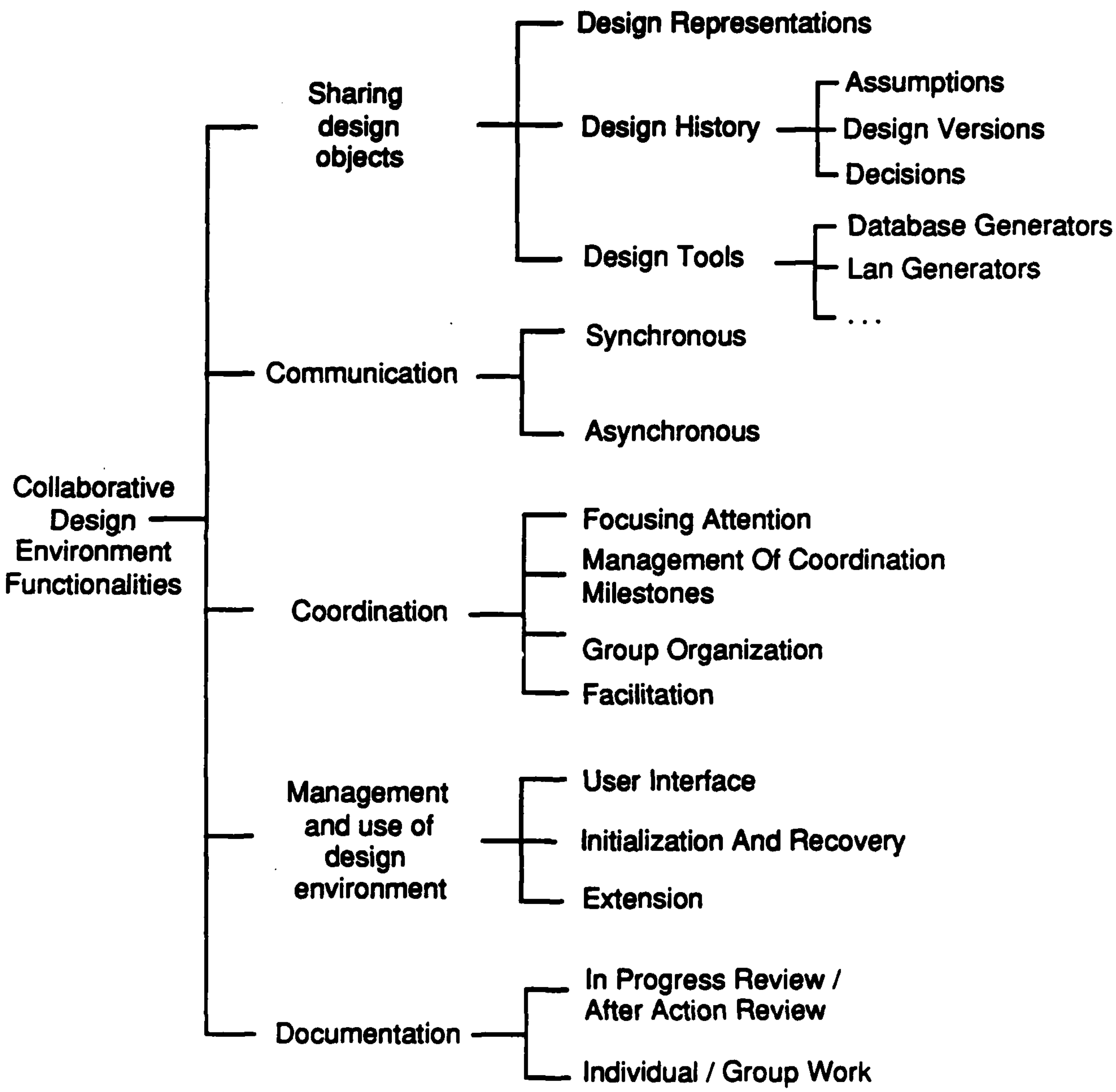


Fig 3. Collaborative Design Environment Environment Functionalities

TABLE 1

Desired Characteristics of Process Modeling and Simulation

- Accept several kinds of inputs for factory simulation including: graphics; high-level language declarations; and program modules written in specific languages
- Combine visual formalisms with executable specifications
- Provide default description of various processes/subprocesses
- Permit explicit description of the key flows and parameters:
  - People] allows description of the organizational model
  - Authority
  - Parts
  - Tools
  - Information] basis for a managerial model
  - Cost
- Allows people/models to be included at various levels within the simulation
- Automatically generate data bases, local area networks and process control software
- Allow systematic application of "downstream" constraints in a "what-if" design process
- Allow progressive top-down refinement of the simulation (higher and lower levels) and supports bottom-up integration (from lower to higher levels)
- Integrate with the rest of the enterprise, thus allowing
  - The concurrent design team to test the manufacturability and cost-effectiveness of the design
  - The factory personnel to integrate the different tools in their ongoing operations

TABLE 2
The Simulation Continuum from A Human-Machine Integration Perspective

Simulation Type / Comparison Criteria	Analytic Simulation	Manned Tabletop Simulation	Reconfigurable Station Simulator	Networked Simulator
Goal	• IRFPA Manufacturing process • Modeling (user perspective) • Function allocation • Information analysis • Workload analysis	• Manufacturing station display • Operator error and error rates assessment • Subjective workload analysis	• Layout optimization • Physical controls and displays prototyping • Training • Stimulus - response compatibility	• "Test drive" integrated processes in manufacturing plant • C3 training • Coordination errors • Communication and coordination load analysis
Technical Basis / Tool	e.g., discrete-event simulation	e.g., discrete-event simulation	Man-in-the-loop Part-task simulator	selective computational fidelity in M-I-L simulation
Operator Representation	Canonical rule-based model	Designer/Analyst plays the operator	Designer/Operator	Operator
Fidelity Available	• Informational • Functional • computational (selective)	• Perceptual • Informational (internal and external) • Display	• Physical • Anthropometric • Informational (internal only) • C&D level	• Informational (external and internal) • C&D level • Physical • Computational • Anthropometric • Environmental
Cost	Very low	Very low	Low - Moderate	Moderate - High

• **Live Video**	—	**promote human-machine relations (cultural change)**
• **Stored Video (compressed in education workstation)**	—	**embedded help and examples in workstation usage and architecture / design tasks**
	—	**footage of typical component assemblies, fab line**
• **Computer Graphics**	—	**interactive modeling and design of product and process (physical abstraction, symbolic metaphor)**
	—	**primary medium of communication among participants**
	—	**display of results (e.g., bar charts, pie charts)**
• **Animation**	—	**dynamic process simulation, alerts in testing, constraint violations**
• **Digitized Voice**	—	**constraint violation alert in hands-free operator environment**
• **Voice**	—	**remote teleconferencing with live video support**

FIGURE 4 Interactive Multi-Media Components and Usage

TABLE 3

Key Capabilities of Interactive Platforms

Category / Topic	IVD	CD-ROMXA	DVI	CD-I	CIG	Optical Disc	Tape
Storage Capability							
• Massive Size	xxx	xx	xxx	xxx	xxx	xxx	xxx
• Data Access	o	x	x	x	xx	x	o
• Data Reliability	xxx	xxx	xxx	xxx	xx	x	xx
• Read / Write	–	–	–	–	xx	1	x
Multimedia Storage							
• Video	xx	o	xxx	x	o	x	x
• Graphics	o	xx	xx	xx	xxx	xxx	x
• 3-D Animation	o	x	xx	x	xxx	xx	x
• Audio	x	x	xxx	xx	xx	x	x
• Text	o	x	x	o	o	x	x
• Database	o	x	x	o	o	x	x
Multimedia Delivery							
• Video	xx	o	xxx	x	o	o	o
• Graphics	o	xx	xx	xx	xxx	o	o
• 3-D Animation	o	x	xx	x	xxx	o	o
• Audio	o	xxx	xxx	xx	xx	o	o
• Text	o	x	x	o	o	o	o
• Database	o	x	x	o	o	o	o

LEGEND:

xxx Eminently suited xx Well Suited x Applicable o Possible, but significant drawbacks

– Not Suited 1 Write once read many (WORM)

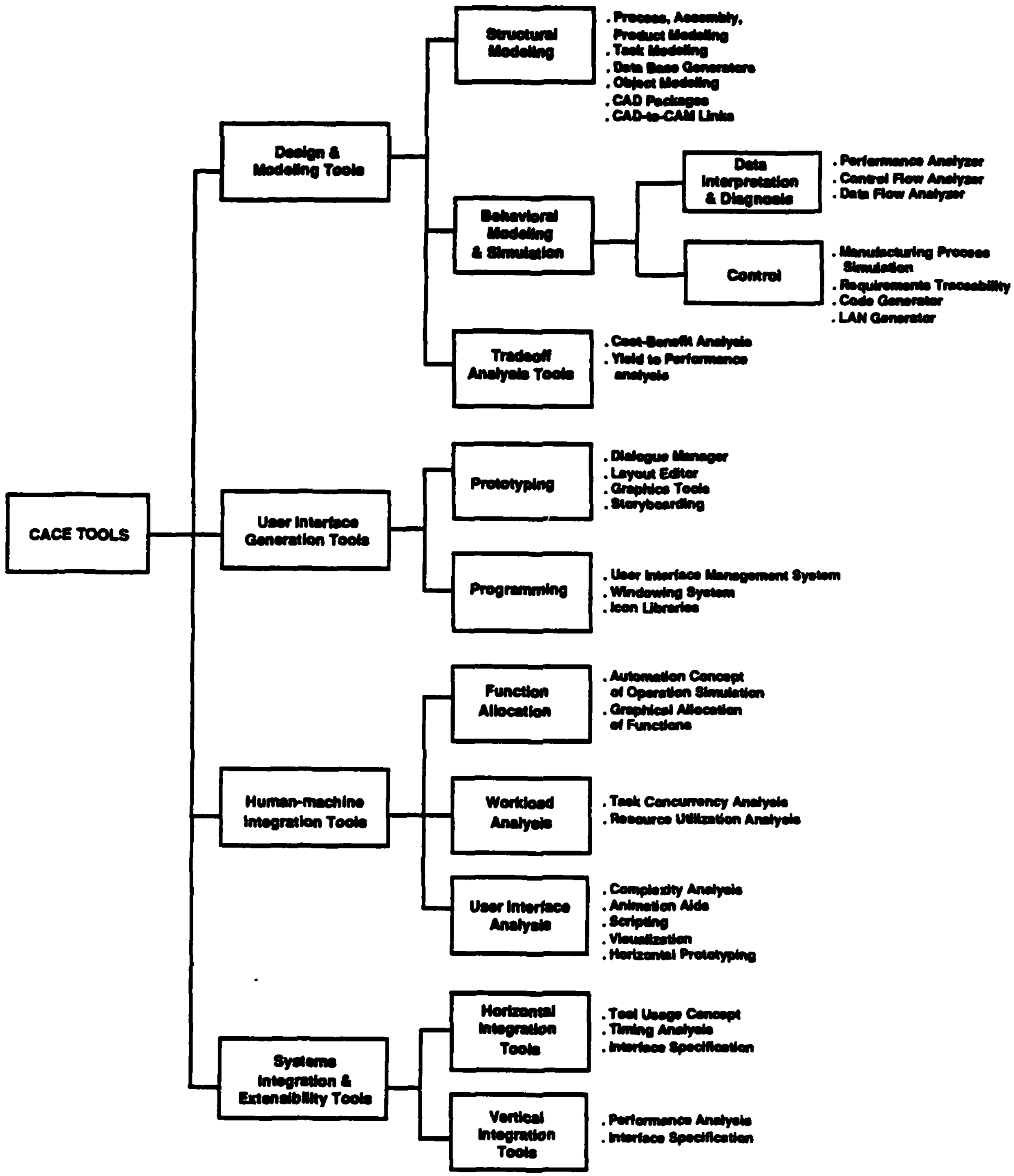


FIGURE 5 Major Elements of CACE

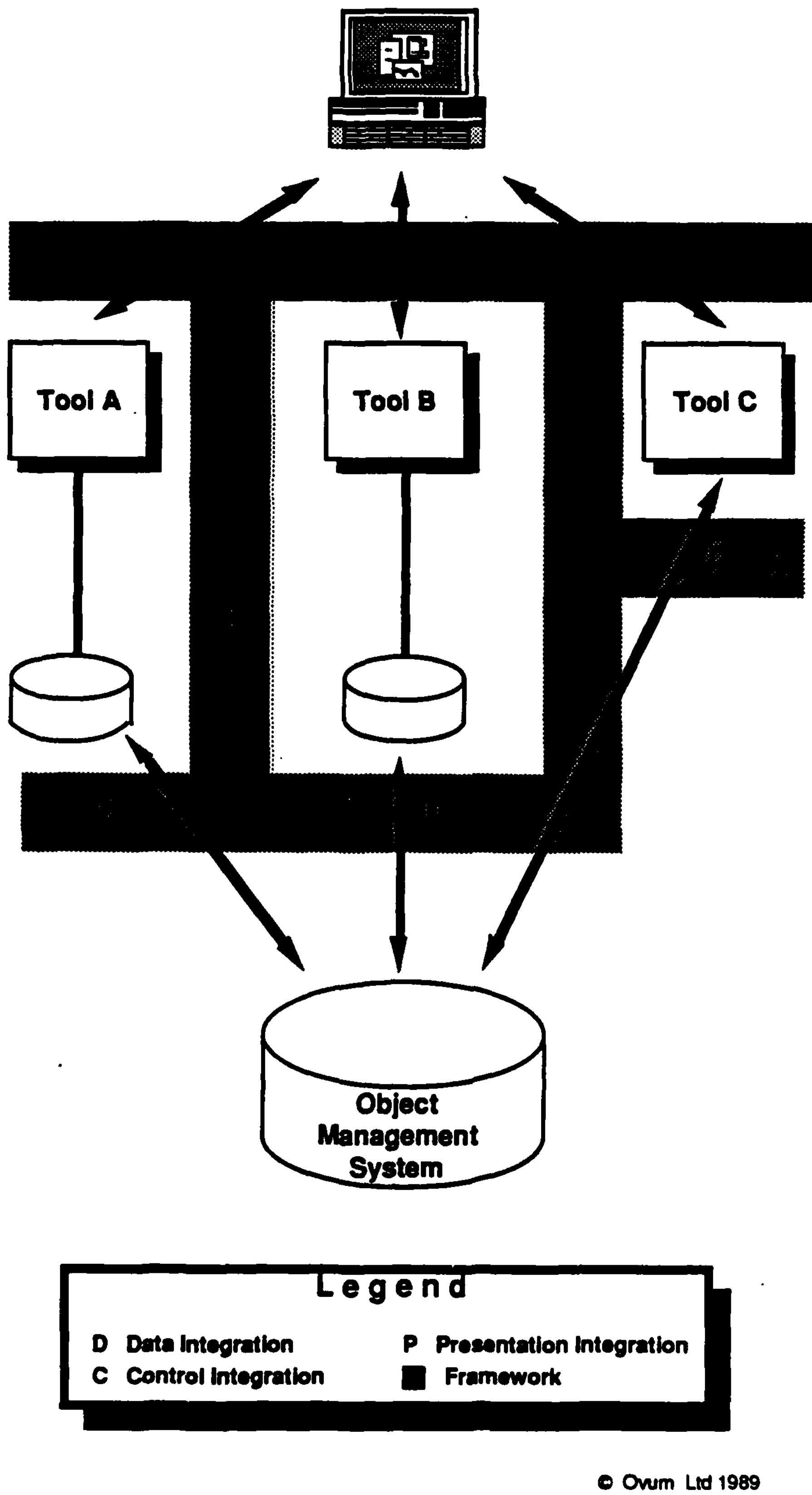


FIGURE 6 Types of Integration

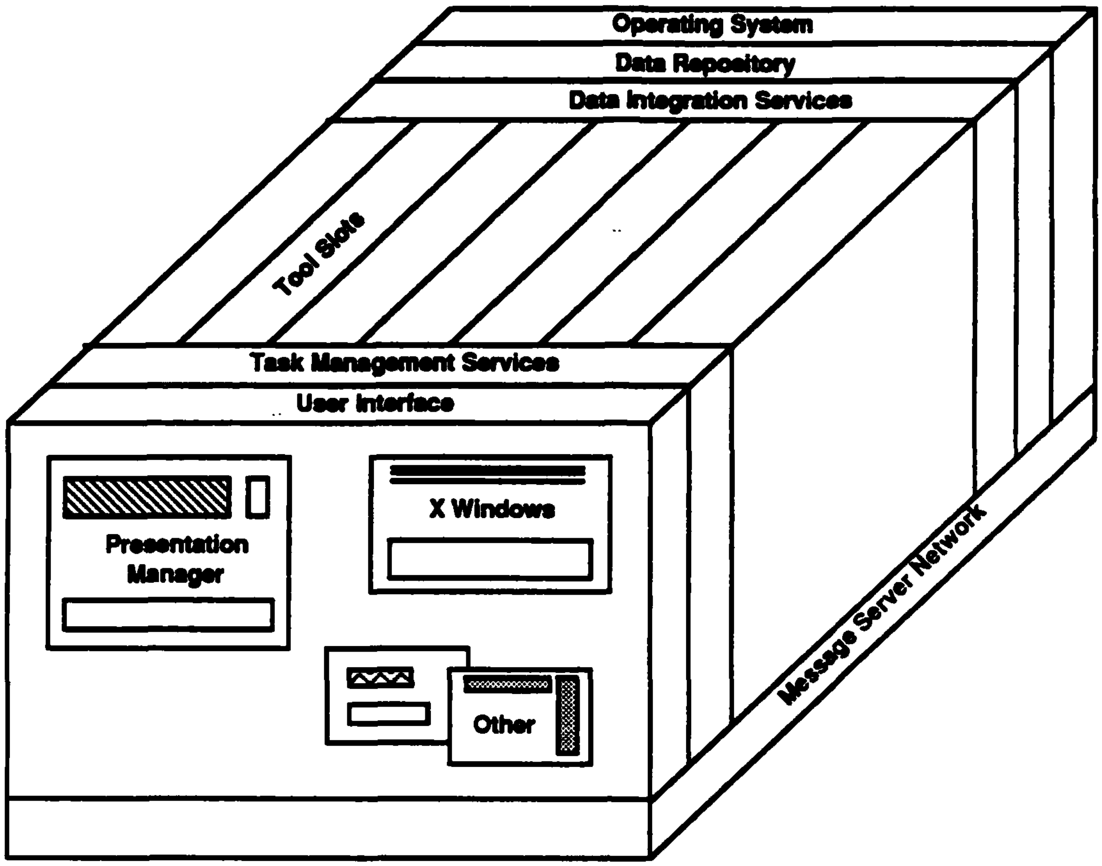


FIGURE 7 Full IPSE Model
(Source: Hewlett-Packard Engineering Systems; as printed in SOFTWARE MAGAZINE, October 1989, Sentry Publishing Co., Inc., Westborough, MA 01581

TABLE 4

STRATEGIES FOR LARGE SCALE IMPLEMENTATION

(1) Demonstrations of the technology for potential users
(2) Group consensus on the goals for the technology
(3) User participation in development of an implementation plan
(4) Implementation on a trial basis in one area of the organization
(5) Evaluation of the implementation trial
(6) Revision of the implementation plan (if needed)
(7) Monitoring and periodic review of the implementation effort

TABLE 5

TAXONOMY OF COOPERATIVE WORK STYLES IN TELECONFERENCING

(1) Individual work versus group interaction.
(2) Anonymity versus signed responses
(3) Feedback of group results versus no feedback.
(4) Aggregated versus unprocessed results.
(5) Voting versus no voting.
(6) Numerical processing versus symbolic processing
(7) Filtered information (selection mechanisms to access only selected items) versus unfiltered information.
(8) Synchronous versus asynchronous interaction.
(9) Sequenced versus free or unstructured interaction.
(10) One-time access to information versus continuous access.
(11) Patterns of communications: one-to-one, one-to-many, many-to-many, many-to-one.

TABLE 6

TRANSITION STRATEGIES

- Start Small, then Expand
- Staged Series of Demonstrations
- Indoctrination of End Users
- Storyboarding
- Horizontal Prototyping
- Vertical Prototyping
- Exploitation of Inplace Procedures and Tools
- End Users Involvement in All Phases

EXAMPLES

OF

TRAINING AND ASSESSMENT TECHNOLOGIES

Assessment of Intelligent Training Technology[1]

Alan Lesgold
University of Pittsburgh
Pittsburgh, Pennsylvania 15260

Over the past decade, there have been considerable research and development in applications of artificial intelligence to education and training. In several cases, training systems have been produced that are receiving practical use. More commonly, so far, managers are starting to face decisions about whether a prototype research system has potential utility. In this chapter, I view the assessment of intelligent training systems from a long-term perspective, and discuss the different kinds of decisions that require assessment of intelligent training technology, and a number of specific assessment issues, considered in light of current theory and experience. In particular, I draw on experiences with the Sherlock coached practice environment for electronics troubleshooting.

Immediate Effectiveness versus Potential

Technology assessment in the world of intelligent training systems must consider not only the effectiveness of a training system but also the likelihood that it can be assimilated by the organizations that could use it. This can be seen either superficially as a marketing problem or more deeply as a problem in changing schooling or training. In either case, though, a product must not only be effective; it must also either fit the existing organizational structure and available technology or it must be so attractive as to bring about adaptive changes that make it useable.

When an artifact is assessed for its immediate utility, evaluation is extremely straightforward. We try it and see how it works. Sometimes we can develop quantitative assessment approaches that allow us to rank alternate products. For example, testing laboratories have mechanical devices that simulate sitting in a chair and getting up. With such devices, we can count how many sitting/rising cycles it takes before a chair's springs fail. With a mixture of such tests, we can develop composite scores that assess the durability of chairs. In other cases, where more subjective judgements of adequacy are involved, we can ask potential consumers to rate properties of a product. So, we see ratings of orange juice, microwave popcorn, wine, and other food products by panels of tasters, either professional or from the lay public. For some products, the effect of using the product can be directly measured. For example, when testing razors, we can count the number of cuts received by a sample of razor users and perhaps even measure the lengths of hair roots left after shaving.

Instructional products are often evaluated the same way. We use the product and measure its positive and negative outcomes. Usually, this is done by testing students after their use of a product and seeing whether they do better on goal-relevant test items after using the product than after some alternative treatment. Costs, corresponding to the razor cuts of the preceding example, are also assessed. Often, there is an explicit comparison of requirements for the instructional treatment being tested to the available resources in some pool of representative schools. Other costs, such as teacher preparation time, student class time, etc., are also considered.

User acceptance is also an important consideration when instructional products are evaluated. For example, certain textbook series are considered to have great instructional potential but to be market risks because teachers won't adopt them. Often, this occurs when a textbook series fails to match current curricular sequencing, requires significant teacher preparation time investment, or fails to provide certain aids to teaching that competitors make available. A cost likely to be associated with a technological aid to education is the investment of time teachers must make to learn how to use it. A particular problem for computer-based systems is whether the computer equipment it requires is broadly available.

In the case of computer-based consumer commodity products, assessment must consider user acceptance and installed platform base. Many a product has failed in the market, even though it was demonstrably better than its competitors, because it required a different operating system or more memory. More trivial sources of product failure include complex copy protection schemes, complexity in providing the right size of diskettes, lack of compatibility of files with extant products in the marketplace, etc. Clearly, from the standpoint of an educational product meant for use today, an assessment of the product must consider not only whether it is effective when used but also whether people are prepared to use it and willing to use it.

When evaluating an educational product's potential for the future, these factors are harder to deal with. The computers that schools have today are primitive and characteristically inadequate for many of the most exciting educational technology possibilities.[2] To restrict positive assessments to software that runs on such machines is foolish, especially when schools show a continuing ability to upgrade their resources, albeit slowly and spasmodically, in response to technological change.[3] However, it is difficult to predict when enabling conditions will arise in any market, including the school market. Fortunes have been made and lost in guessing whether DOS, Unix, Windows, or OS/2 will prevail, for example. Accordingly, it seems very important at least to understand what conditions will be required for a promising prototype system to be used and valued, and it would be extremely helpful to have means of assessing the likelihood of those conditions arising.

In the case of intelligent training systems, some information is available concerning the hardware platforms that will be prevalent in the future, and there is also information about migration tools that might broaden the range of alternative futures for which a given system might be adapted. For example, it is clear that Windows 3.0 is a major force in the computer world. Combined with other forces previously evidenced, Windows 3.0 assures that there will be a relatively large population of computers in the business and industry world that have 80386SX, 80386, or 80486 processors; four or more megabytes of memory; and decent (VGA or better) graphics.

On the software side, much of the research base in artificial intelligence depends upon Common Lisp running on large Unix work stations. Currently, delivered systems are often rewritten in C so they will run faster on standard work stations (major Lisp vendors are working to complete adequate delivery systems for Lisp that run efficiently on multiple platforms). Rather recently, Smalltalk has emerged as a programming language of choice for training system development, and it is what my own team uses for its biggest and most applied project. Recent actions in the commercial software world suggest a move toward use of object-oriented languages such as Smalltalk because they produce software that is more maintainable. I discuss these issues in more detail later in this chapter.

In the easy case, then, a decision-maker can predict, with a high degree of confidence, that the hardware and the software architecture required to support a desirable prototype system will be in place. Often, however, given the requirements of development, there will be a gap between the hardware platform for a prototype system and the hardware broadly available in the short term. A conservative view of the problem of assessing both software quality and the likelihood of a hardware plant to run the software would be that only software that makes a striking contribution will succeed in navigating the hardware gap. From this viewpoint, assessment means a search for evidence that a product not only is effective but further that it is so effective that it cannot easily be avoided. A global finding of massive effect, though not particularly informative for product refinement, provides a basis for predicting that a product will be used, even if it requires improved hardware. More broadly, the bigger the infrastructural investment that would be needed to use a piece of software, the bigger the demonstration of general efficacy needs to be.[4]

These observations are informed by our own experience with the "Sherlock Project," a long-term effort to shape a technology of coached practice environments for training complex problem solving jobs in the Air Force. Our first "product," Sherlock I, while it needed many refinements, did produce the massive effect needed to assure likely adoption, and led to support for ambitious efforts to migrate, refine, and generalize the system. Consequently, the way in which we evaluated it may be worthy of examination. Below, I discuss this project, especially our approach to assessing it. I conclude by reflecting on some issues critical to assessment of training technology that became apparent in the course of the development of Sherlock.

Assessment Experiences from the Sherlock Project

What is Sherlock like? Since 1983, my associates and I have been developing a technology for training complex problem solving jobs for the Air Force. The Sherlock Project has been an extended effort to develop improved approaches to the training of complex job skills, especially training that supports transfer to new but related jobs. After empirical studies of experts and trainees, the work has revolved around a family of prototype training systems called Sherlock. Two generations of coached intelligent apprenticeship environments have now been built. The first of these, Sherlock I, was evaluated extensively in the field. Its successor, Sherlock II, is largely a response to lessons we learned from the evaluations. Sherlock II includes a simulation of a very complex electronic device, a "test station"[5] with thousands of parts, simulated measurement devices for "testing" the simulated station, a problem selection scheme that presents fault diagnosis problems within the device simulation, a coach that provides assistance when help is requested in the course of diagnosis, and a reflective follow-up facility that permits a trainee to review his performance on a problem and compare it to that of an expert.

In developing this system, we had a number of psychological concerns. We wanted to afford opportunities for learning by doing, and this required simulation of complex job situations, not just small devices. Also, we wanted to tailor the coaching provided by the system to the knowledge needs of the trainee. Ideally, the trainee should be kept in the position of almost knowing what to do but having to stretch his knowledge just a little in order to keep going. Consequently, hints should be provided with some inertia. They should provide enough help to avoid total impasse but this help should come slowly enough so that it is easier to think a bit on one's own than to wait for the correct next step to be stated completely. This requires considerable modeling of student capability and may also require modeling of the course of trainee-machine interaction. Of course, it also requires expert modeling, in order to know what advice to give.

During the course of difficult work, it is not likely that much extended learning can take place. Both psychological experimentation (cf. Owen & Sweller, 1985; Sweller, 1988; Sweller & Cooper, 1985) and theoretical models of case-based learning (e.g., Mitchell, Keller, & Kedar-Cabelli, 1986) indicate that learning from task situations requires a lot of cognitive effort. For this reason, we have put much of the power of our training systems in post-problem reflective follow-up. After solving a problem, which requires about a half hour of effort, trainees can review their actions step by step, asking what an expert would do at any point along the way. Or, they can simply ask for a trace of an overall expert solution. At each step in either trace, they can receive background information on why the step is being taken. Also, color diagrams at each step in the trace show what is known about different parts of the system being diagnosed (parts proven good in green; paths with incorrect information in them in red, etc.). Much of this capability rests on the same knowledge structures that support coaching during problem solution.

Sherlock I ran on a special artificial intelligence work station and was written in Interlisp and Loops, a proprietary object-based language for Interlisp. Trainees interacted with the system by pointing to screen-graphic renderings of the front panels of devices and to schematic diagrams of their innards. Sherlock II runs on a standard 80386 machine, requires around 8MB (depending on the version used and research requirements for data collection), about 20MB of hard disk, a processor speed of 20MHz or higher, a videodisc player and interface, and a GENLOCK card to mix the video and computer graphics images. Trainees interact with Sherlock II by making selections from menus and by pointing to video views of test station components. For example, to make a resistance measurement, a trainee would mouse on a screen icon of the hand-held digital multimeter to access the front panel display of the meter in video, indicate knob settings on the video image with the mouse, select the component he[6] or she wanted to test from a menu, and then indicate meter probe locations by pointing to a video image of the component. Coaching advice also is available via the menu system. Sherlock I, in spite of its more specialized artificial intelligence environment, was much less intelligent and more brittle in its knowledge than Sherlock II. In addition, it lacked the reflective follow-up capability that we now believe to be of great importance.

Basic assessment results for Sherlock I: The massive effect. Sherlock II is just now being completed. We do have field assessment data for Sherlock I. Overall, Sherlock I was excessively rigid but remarkably successful. Because the goal structures *for each problem* were hand coded and many hints were specific statements written in advance, the system was not very extensible, and the interaction between trainee and machine somewhat rigid. However, the bottom line is that 20-25 hours of Sherlock I practice time for trainees in their first four years of duty produced improvements almost to the level of their more senior colleagues who had many more years on the job. In more practical terms, trainees could not generally troubleshoot test station failures before the Sherlock training, but they could afterwards. Here is how this conclusion was established.

The goal of Sherlock I was to train the ability to diagnose test station faults, the hardest part of the F-15 manual test station job. So, we[7] used simulated test station diagnosis problems in the field evaluation (see Nichols et al., in preparation, or Gott, 1989, for details). Virtually all of the job incumbents at two Air Force bases participated in the evaluation, a total of 32 airmen in their first four-year term and 13 at more advanced levels. The 32 first-termers were split into an experimental and a control group of 16 each. The experimental group had a mean experience of 28 months in the Air Force, while the control group had a mean of 37 months. The advanced group had about six more years of experience, a mean of 114 months. The simulated test station diagnosis problems were presented verbally. An example problem is the following:

> While running a Video Control Panel unit, Test Step 3.e fails. The panel lamps do not illuminate. All previous test steps have passed.

In such a situation, the unit being tested is usually defective, but in this particular case, it turns out that a relay card in the test station is bad. This type of problem is one that is extremely difficult for novice technicians. In the evaluation study, the technician would hear the problem statement and then be asked two questions: *What would you do next?* and *Why would you do that?* He would then be told the result of his action and the cycle of questioning would repeat until the problem was solved or an impasse was reached. Verbal protocols of these problem solving interviews were then given to Air Force experts who scored them blind to the condition assignments of the subjects. Scoring scales were derived from expert rankings of the problem solving performances using "policy capturing" techniques (discussed below).

Figure 1 shows the results. Given group standard deviations ranging from 12 to 29, the results show that the experimental and control group pretest means and the control group posttest mean are at one level and the experimental posttest mean and the advanced group mean are at a second, much higher, level. The data were also considered from a second viewpoint, the amount of on-the-job experience needed to produce gains equivalent to those produced by the Sherlock experience of 20-25 hours of coached practice. The totality of the pretest data was used to generate a regression coefficient for predicting months of Air Force job experience from these test scores. Using the scale created by this regression, the gain shown by the experimental group from pretest to posttest is equivalent to about four years of job experience, using conservative estimates, although the confidence interval for this estimate is necessarily huge, given the small numbers of subjects. A follow-up testing six months after training showed retention of over 90% of the gains made from pre to post testing. Overall, then, Sherlock was very successful, in terms of producing the ability to do the specific job of manual F-15 test station diagnosis, which is not readily acquired from simple on-the-job experience or from the training now available prior to reporting for work.

Interpreting the massive effect. The massive effect of Sherlock is real, but it requires interpretation. Basically, Sherlock is effective because it has no competition. That is, it affords opportunities for learning that were not available before, either on the job or in the classroom. A variety of logistic and other limitations make it very difficult for on-the-job training to be effective in teaching the hardest parts of the job. When those hard tasks appear, it is often an urgent situation, in which training opportunities take a back seat to the need to get equipment back on line. Further, the specific events that would present good training opportunities with respect to our high threshold criterion are relatively rare. In the schoolhouse, it is even more difficult to create realistic situations in which the hardest diagnostic tasks could be practiced, and such practice seems to be critical to attaining high levels of skill.

When looking at high-end job performance, even in purely cognitive tasks like fault diagnosis, there often is no substitute for realistic practice. Abstract principles can be taught in the classroom, but it is rare for those principles to automatically be applied in the real world without some practice, especially in situations of great complexity and likely stress. Consequently, one interpretation of the massive effect of Sherlock I is that it provides practice, under simulated task conditions, that is sufficient for learning, whereas prior learning opportunities within on-the-job practice did not. Seen from this viewpoint, the assessment issues that are raised are somewhat different in character. We consider each of these, in the context of the Sherlock experience, in the next section.

Issues in Intelligent Training Technology Assessment

Reality and Nature of the Main Effect

The first issue is the reality of the main effect. If one is going to assert that a system accomplishes a major chore that is otherwise not readily accomplished, this needs to be backed up. It is necessary to show that the system produces levels of competence that are worth more than the levels of competence achieved without using the system. At one level, this was easy. We could document the costs of not having technicians who could troubleshoot test station failures, and we were able to show that after Sherlock training, technicians could handle problems that they could not handle before. Since the Air Force was hiring civilian experts to stand by to assist uniformed personnel when test station faults occurred, the costs could be documented. However, we feel it important to have a quantitative documentation of the training outcome as well as of the monetary value of the training. That is, we wanted to have quantitative scores of performance in the test station troubleshooting tasks whose value we had established.

To get this, we had two basic options. On the one hand, we could list a number of properties of good test station troubleshooting activity and then score troubleshooting episodes for the presence or absence of those properties. For example, making measurements instead of swapping components is an important property. We could decide arbitrarily to subtract some number of points from a person's score for each action required to solve a problem, charging more for swapping than for testing. Further, we could almost certainly reduce the costs to dollars. Each action takes time, which has measurable cost, and swapping a good part generates a variety of measurable costs. Depleting an inventory has at least indirect costs, since the inventory must be kept higher if such depletions must be anticipated.

It is also important to note that Sherlock I was based upon many design principles, some of which were enumerated above. Which design principles have what effect on the various possible outcome measures is difficult to establish. One implication of this difficulty in isolating the mapping of treatment parameters to effects is that once a main effect is established, detailed parametric research may be justified. In the case of Sherlock, the initial results have motivated support for just such a program of research, which is just now beginning in the context of Sherlock II.

Transfer

On the other hand, we could have experts evaluate the performances of those who did or did not use the system, and then compare their ratings. At first blush, this seems considerably less satisfactory than the quantitative approach just described, and it would be if we were concerned only with immediate costs and performances. However, if we want to predict transfer to systems that may not yet exist or that may not be available for testing, especially when predictive studies of transfer may not have been carried out for the knowledge domain, then it may be necessary to rely on expert appraisals of competence. Expert judgements have known shortcomings. In the military context, it is not unusual for staff who are cooperative to be valued over those who are not, independent of their expertise, for example. With respect to changes that we expect a tutoring system to introduce, such social factors represent error variance in an evaluation design.

The Air Force worked with us to find ways to use expert judgements to derive scoreable properties. Nichols et al. (in press) applied techniques of policy capturing to develop scoring schemes for the troubleshooting task we used in evaluating Sherlock I. The basic scheme is very straightforward. Difficult troubleshooting problems are posed verbally to the testee. Testees call for various actions to be performed in solving the problem. Results of each action are provided to the testee. A trace is kept of all the activity of each trainee in solving each criterion troubleshooting problem. Experts are asked to examine the traces and to rank order them. Then, they are questioned about the bases for their rankings. From this questioning, one derives a list of features that contribute to judgements of expertise. The next step is to assign point values to each of these features, so that they can be used to set a score for the trainee. Finally, the scoring scheme thus developed can then be used to score additional performances by the same or other test subjects.

Setting the point values for each scoring feature can be done several different ways. One approach is to set the point values to best predict the rank orderings assigned by experts to the whole performances. This is a common strategy and generally involves regression analyses. The strategy makes sense if we assume that the experts are making their holistic judgements solely with respect to issues of expertise we care about. Sometimes this is not the case. A second approach is to analyze the features cited by experts and to evaluate them. This can be done both empirically and theoretically. Empirically, it is possible to do clustering analyses of the scoring features. Once clusters are identified, some amount of theorizing is required to decide why certain features cluster. This may result in a decision that some features are good indicators of important job knowledge while others are mainly generic indicators of being "good soldiers." For example, following recommended troubleshooting sequences religiously may be highly valued but may not correlate with indicators of deep domain knowledge.

One issue that arises when the policy capturing approach is used is that there is potential loss of explainability for scores. Saying that a trainee got a high score because he did things that experts value highly is not quite as satisfactory as saying that the high score came because performance approximated that of an expert in identifiable ways. For this reason, we are moving toward a modification of the pure policy capturing approach. We are now experimenting with an on-line evaluation scheme in which our expert model directly tallies various indications of expert-like and non-expert-like actions. These tallies can then be used in two ways. First, they can be summarized in ways that match the abstractions implicit in the expert model's goal hierarchies for performance. Second, regression analyses[8] can be done to predict human expert judgements of overall performance from the more microscopic objective tallies.

Nichols et al. (in press) probed for data beyond problem solving steps. Test subjects were asked to explain each step that they were taking, to indicate expected outcomes of tests, and to comment on what was learned from each new piece of information (such as a measurement made on the circuit). This approach was useful in providing information about performances that had high face validity. People who offer principled explanations for their problem solving actions and who solve the problems efficiently are certainly candidates for good transfer performance. On the other hand, the very requirement of explaining steps that are being taken may produce performances that are more systematic, and even more generally expert, than otherwise would occur. The requirement to explain one's performances is a significant intrusion into task performances. So far, there is no indication that it distorts results, but further study seems advisable.

Overall, it seems well advised to ground the evaluation of a training system in performances of difficult jobs from the target task domain.

Mode of Presentation

In the first field tests of Sherlock I, some people in our client organization objected to using computer-administered problems for assessment, because the training program provided practice in using our computer system as well as job-specific practice. For the most part, this objection is attenuated as the evaluation becomes more heavily grounded in details of expert-like performance. Still, it is surely possible that a testee unfamiliar with the interface might appear to be performing less optimally than would otherwise be the case. For example, it is possible to "miss" in making choices from a pop-up menu, and such a "miss" might result in a tally of a non-optimal choice during problem solution. Of course, the verbal problem simulation approach we have used so far in field evaluations is also problematic, since it requires trainees not only to be able to solve problems but also to be able to articulate their decision processes. However, the biases due to verbal aptitude requirements when verbal simulations of problem solving are used apply to both experimental and control conditions, while the biases due to computer-administered simulated problems apply only to the experimental group. Accordingly, we recommend that post-testing not be done via computer unless both experimental and control subjects are well trained in the interface.

It can still be useful to use the computer for administering and scoring the problems, however. The experimenter can sit at the computer and enact each action of the trainee. Also, by modifying the system to accept annotations along with actions, it becomes possible to log the trainee's explanations along with the trace of actions. A major advantage of this approach is that testing need not be done by a domain expert, since the system simulation in a training system like Sherlock will be able to provide the result of every action the testee proposes. In earlier field work, we occasionally had testees call for an action whose results the tester could not determine. Since some of our testing was done on third shift, this meant that an expert had to be phoned in the middle of the night to determine what meter reading to report back to the testee in response to the action proposed. Given that systems like Sherlock contain verified device models, a tester could use those models to guide his or her interactions in verbally posed problem sessions.

Ideally, systems will be developed for multiple jobs in a "job family." Again, once testees are familiar with the basic interface conventions used for a family of training systems, the systems themselves make good testing and data logging systems. Transfer can be assessed by noting the trajectory of performance over training sessions for both original training on one system and transfer training on a second. Given proper counter-balancing of order of training, it is possible to quantify transfer in terms of time (and its monetary value) saved in learning the second system given training on the first. Again, there are objections to using the training system as an assessment vehicle if prior experience with the interface is not controlled, and again the systems may make good examiner stations even when testing is done via verbal interactions.

Assessment Advantages of Object-Based Approaches

Object based design has power that is making it the standard in the software development world. For the very same reasons, object based design may be key both in training for transfer and in assessment of transfer potential. More generally, intelligent training systems offer a peculiarly good opportunity for achieving synthesis of learning and assessment, which has been widely advocated as a goal for education and training generally.

The expertise in systems like Sherlock II is represented in computational "objects." An object is an independent piece of computer program that stores its own local data and can thus respond to various requests that other parts of the system might make of it. Knowledge about how to deal with specific components of the system being diagnosed is embedded in the objects that represent those components. For example, a component in a system will be represented by an object, and component objects will have routines relevant for the objects they represent. The object for a given component, such as a particular type of printed circuit card, might "know" how to draw the component it represents on the screen. It also will likely know how to model its component as part of an electronic circuit, how to test that component, and how to coach the trainee in his interactions with that component. Since most training systems will do some sort of student modeling, the object for a component also needs to know how to record the student's interactions with that object, how to score those interactions, and how to make the recorded information available to other parts of the system that might be generating more abstracted evaluations of the student.

Further, objects are generally arranged in an inheritance hierarchy to facilitate system development. General objects are defined that have basic capabilities. When more specialized capabilities are needed, new objects are created that inherit general capability from their "parents" but add to that some specific functionality. For example, all component objects may need to tally the time of each action taken by a trainee that involves the component within their "scope." The time tallying routine would, in a good object-oriented programming system, be defined only once, for a general component object, and the routine would be inherited by more specialized objects that also require it.

At a higher level of expert function, a level that abstracts over specific components and even types of components, expertise is represented as a goal structure for problem solving. In our approach, a task analysis will lead directly to specifications for the computational objects to be used in training systems. The goals of training are represented by objects, and those objects, too, must be able to perform, coach, and illustrate their specific bits of expertise. Further, they, too, can be readily modified to keep track of how well the trainee approximates expertise in his or her performance.

A goal hierarchy represents one kind of abstraction of expertise, but there is also another kind. The concept of an inheritance hierarchy of knowledge, which is central to object-oriented programming (though lacking in some partial object-oriented programming languages!), also has potential for the prediction, training, and assessment of transfer, both potential and actual. This is especially the case when specialized objects do their work through a combination of a few specific actions combined with a "call" to a more generic routine. I currently believe that analyses of multiple jobs for transfer should result in products that specify inheritance hierarchies of knowledge components. Further, I believe that transfer can then be predicted by noting the relative amounts of knowledge that are shared, that require only minor specializations of what is already known, or that require major new learning. I am currently trying to refine such an approach to task analysis.

Given the decision that task analyses will lead directly to specifications for the computational objects to be used in training systems, and given the scheme of having those objects able not only to simulate expertise and train, but also to assess trainee performance, it is a small step to systems that can assess transfer potential as well as job-specific learning as they train. The step entails objects' contributing to decisions about transfer.[9] For example, if there are cases where performance from the viewpoint of a specific object is expert-like, while performance from a more general viewpoint is not, this may indicate that the trainee has learned a successful algorithm that does not adequately generalize. Such an indication would count against any claims that the system teaches for transfer. For example, when a technician tests a component by verifying its inputs and then verifying its outputs, or vice versa, he is exercising a general capability that should transfer to many components of many systems. If instead he traces those input-to-output paths that are relevant to a particular situation, he will still be able to decide whether the component is good, but his performance does not have the same guarantee of transferability.

Architectural aspects, and in particular use of object-based approaches, are considerations that should enter into assessments of software for integrated training and testing.

Durability of Effects

For the effects of intelligent training systems to be worthwhile, competence must not only develop and transfer, it must also persist. Training that is durable is much more valuable than training that requires regular "maintenance." While complex, time-sensitive performances inevitably require maintenance, basic competences generally should not, provided that they have been adequately taught and learned. Still, the training world is replete with courses taught, exams passed, predictable and over-rehearsed performances carried out, with minimal long-term effect. Consequently, we felt it important to do retention testing of the Sherlock I system. In fact, retention was extremely high, in excess of 90%. Further, it was relatively uniform--few trainees showed precipitous drops in performance. We think such demonstrations are important, given the cost of intelligent system research and development.

Just as with transfer, it would be helpful to have predictors of retention. However, the development cycle for intelligent training systems is long even without adding extra months to separate initial from retention testing. I propose that instead, once a particular technology for training system development is well established, it would be possible to predict retention from the many micro-measures of competence that are used for assessing competence and transfer potential. In the Sherlock work, for example, if retention rests on conceptual understanding, then those specific performance indicators that depend upon understanding the device being diagnosed and the methods of diagnosis should be good indicators of retention. If understanding plus efficiency of certain basic cognitive processes is needed, then indicators of both should be good predictors of retention. We did not conduct the needed analyses to support this approach when we field tested Sherlock I, but we expect to be able to with Sherlock II.

Predicted retention is not the same as actual retention, especially when prediction formulas from one training system's history are used to estimate likely retention for a different system, but it is a start. To the extent possible, predicted retention measures should be supplemented by actual measures, at least on an occasional audit basis. However, predicted retention is an important aspect of assessment, especially because of the increased emphasis on rapid prototyping and more efficient development of computer-based training. Just as with conventional training approaches, it is always possible to audit performance with the system already in place and in use. Predicted retention would allow first tryouts of the system to be more informative, so that sound deployment decisions could be made earlier in the development cycle.

Boundary Conditions

Like any treatment, intelligent training systems will work only within certain boundary conditions. However, assessments of training seldom attend to boundary conditions. It is common to provide background data on the sample that was used to assess system capability. However, users are generally left to make their own inferences about how and when results from that sample might generalize. Some improvement over that state of affairs is possible and important. Consider the field testing of Sherlock I, for example. It was tested at operational Air Force bases, not at the training school. As a result, it could be assumed that a variety of basic electronics principles and procedures had been mastered by the test sample. Otherwise, they would not have been allowed job site roles. On the other hand, because they were on the job, they had daily concrete experience with the artifacts being simulated. It is conceivable that Sherlock I might not work well in the tech school environment. On the other hand, the scheme in which coaching is always available, so that true performance impasses are unlikely, seems rather robust and might be sufficient to permit a much wider range of trainees to use the system effectively.

From a research and development point of view, it seems appropriate to conduct, for each new design approach to intelligent training system design, specific studies of the boundary conditions under which the system works. This is not too unreasonable. It simply means keeping records and analyzing for data patterns as the first implementations of the approach are carried into broad use. Then, if boundary limitations emerge, they can be considered by future users.

Related to boundary conditions are interactions between components of the instructional treatment and trainees' prior knowledge. Consider, for example, the range of treatment possibilities present in Sherlock II or being considered for future versions. These include practice in solving problems presented according to a fixed progression, a student-model-determined progression, or a student-request-determined progression; opportunities to replay one's own or an expert solution to the problem just solved; opportunities or requirement to critique one's own, or a peer's, performance on a problem; and opportunities for any of the above activities as a member of a cooperating peer group. It is quite possible that some of these learning opportunities might work better for one type of student while others might work better for another. For example, there may be some threshold of domain knowledge that is required before one can frame effective questions about an expert performance. If so, then students yet to reach that threshold may not find opportunities to replay problem solutions and ask questions about them to be very helpful. A second possibility is that some students may have more successful practice with one or another of the approaches and therefore value it more.

Although much about the effectiveness of training approaches for trainees with differing prior knowledge remains to be elucidated, it seems appropriate for an assessment of a training system to consider what is known, and to take such information into account in interpreting the adequacy of the test samples used for evaluating the system. A further step is possible when a system records a detailed trace of system usage by the student. Then, it is possible, by examining the course of learning for specific members of the test sample, to use extant knowledge to make predictions about interactions of prior knowledge with treatment, and to proceed to partial verification of those predictions.

Maintainability and Extensibility

In addition to its utility in promoting learning, an intelligent training system must also be evaluated as a technological artifact. Too often, the process whereby training software is procured results in systems that cannot be maintained or extended. Initially, it will seem to the system purchaser that if the system works, there is nothing else to worry about. However, virtually all training systems require modifications over their lifetimes. Weaknesses in the training are discovered. Devices that are part of the task domain are modified or replaced. New duties are added to the job. New trainees, with different entering capabilities, become part of the training population. All of these things happen routinely in the training world.

The requirements for maintainable software are well-known. The software must be modular, with each module carefully documented. Detailed explanations and specifications of how the system works, and why, should be available.

Standard computer languages should be used and only standard hardware should be required. We have found that a deeply object-oriented approach helps greatly in maintaining and extending software; indeed, we have had to rewrite virtually all code that did not reflect deep understanding of object-oriented practices.

Good object-oriented designs make use of inheritance, so that each knowledge component of the system is defined only once. This facilitates repairs and changes. In the object-oriented paradigm, actions are taken by passing a message to the object that is to act. When, in contrast, actions are taken directly, whenever a change is required, each instance of such actions must be detected and modified. This is very costly. Good object-based design, therefore, enhances productivity immeasurably. For example, having developed software that could display electronic circuit diagrams, with color coding of component states and explanation of components when their diagram representations are "moused," our main software designer, Edward Hughes, was able to build a new system for inputting, displaying, and allowing interactions with troubleshooting flowcharts. He simply made a "flowchart editor object" that inherited most of its capabilities from the same "graphic editor object" that supports the "circuit editor object" he had programmed earlier. At a more mundane level, when I wanted to build a database of hints generated by Sherlock, all I had to do was modify the "hint display object" to dump all text to a file in addition to sending it to the relevant text window pane object.

Just as with any system, the learning time it takes a user to use a software development approach must be considered. Experience to date has been that object-oriented programming skill takes a long time to train--three to six months for a top-notch programmer with good formal computer science background. However, even at that high cost, the approach pays. From a software evaluation viewpoint, use of good object-oriented techniques, including the representation of core design principles and approaches in separable objects, makes a training system much more valuable, and makes changes in the system much more efficient. Moreover, "user" acceptability is very high. Our experience, with several different object-based languages, suggests that it would be difficult to wean software developers away from an object-based approach once they have practiced it.

Conclusion

The approach I have taken in this chapter is to view training system assessment from a long-term perspective. Therefore, I have assumed that transfer and retention, not just immediate rote performance of algorithms, is the key to a system's value. Similarly, I have assumed that modifiability and principled design are also highly to be valued. Surely, there are training systems for which these requirements may be excessive. However, for significant training of personnel for enduring organizations that perform work involving high levels of knowledge and cognitive skill, this "high road" will surely pay. Regrettably, it has not been followed sufficiently often. One of the reasons that training software is seen as extremely expensive is that it often is rigid and narrow, both in its realization as software and in its effects in promoting learning. It is unlikely that this "low road" approach will ever be more efficient at what it does than the human-centered alternative of quickly cobbling together stand-up lectures and mastery quizzes. I believe that a much more effective long-term strategy would be for workers to acquire some of the knowledge they need to work intelligently and adaptively through apprenticeship experiences simulated by intelligent and adaptive software.

References

Gott, S. P. (1989). Apprenticeship instruction for real world cognitive tasks." In E. Z. Rothkopf (Ed.) *Review of Research in Education.* Vol. XV. Washington, DC: AERA.

Lajoie, S., & Lesgold, A. (1990). Apprenticeship training in the workplace: Computer coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning, 3,* 7-28.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1,* 47-80.

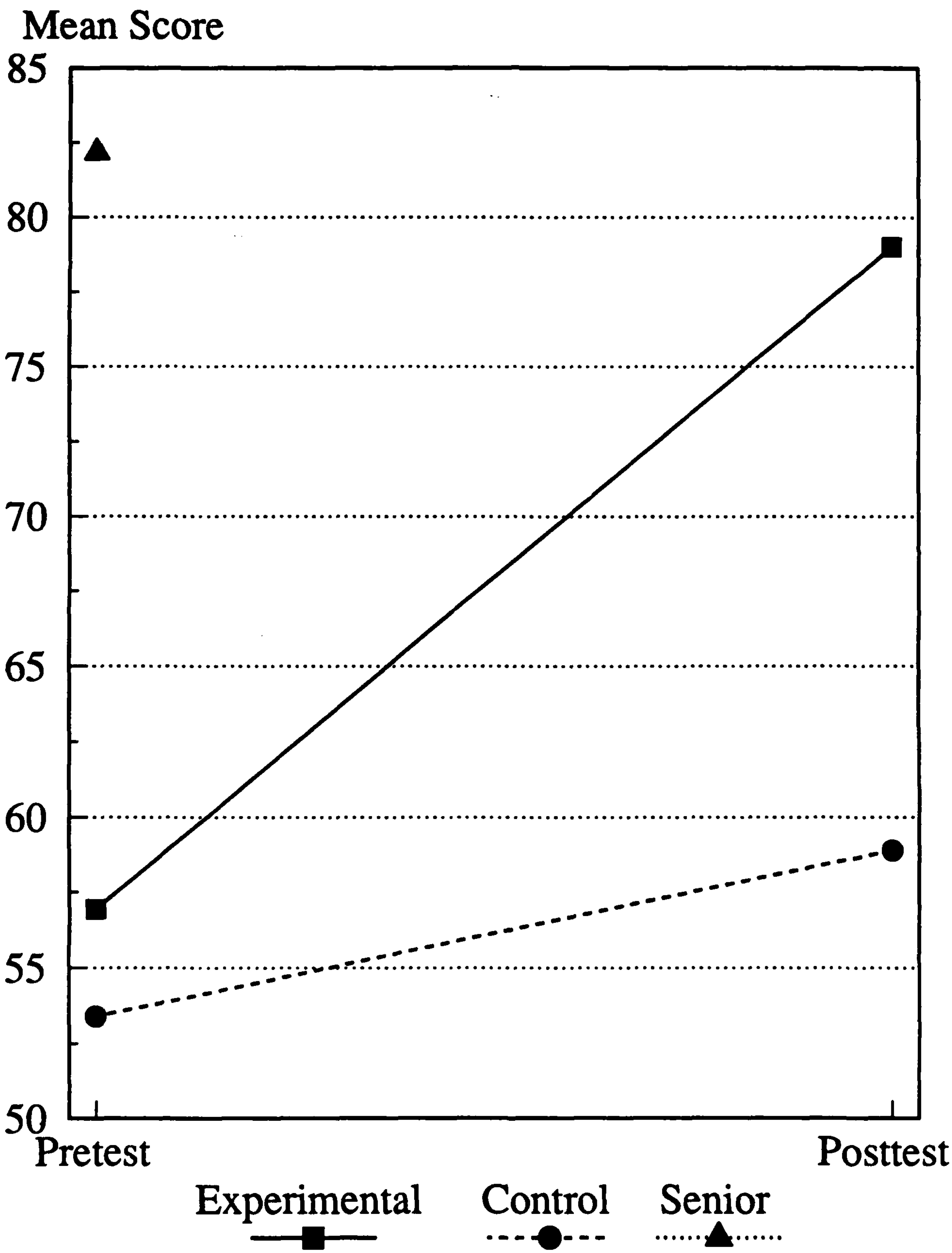
Nichols, P., Pokorny, R., Jones, G., Gott, S. P., & Alley, W. E. (in press). *Evaluation of an avionics troubleshooting tutoring system.* Special Report. Brooks AFB, TX: Air Force Human Resources Laboratory.

Owen, E., & Sweller, J. (1985). What do students learn while solving mathematics problems? *Journal of Educational Psychology, 77,* 272-284.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12,* 257-285.

Sweller, J., & Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2,* 59-89.

FIGURE 1. Results of Sherlock I Evaluation by the Air Force



Footnotes

1. The Sherlock project, from which my ideas come, is the joint work of Marilyn Bunzo, Gary Eggan, Robert Glaser, Maria Gordin, Linda Greenberg, Edward Hughes, Sandra Katz, Susanne Lajoie, Alan Lesgold, Tom McGinnis, Rudianto Prabowo, Rose Rosenfeld, Arlene Weiner, and a number of other colleagues, past and present, at the Learning Research and Development Center, along with Sherrie Gott, Robert Pokorny, Ellen Hall, Dennis Collins and others at the Air Force Human Resources Laboratory. AFHRL supported the work but does not necessarily endorse the statements made in this chapter.

2. The situation for the training world is often similar, though the exact current technology differs. Training directors would love to have highly adaptive intelligent software that will run on 640K DOS machines with 20 MB disks. However, because there are more direct economic forces in the business world than in the education world, training directors will generally buy new equipment if it proven to be cost-effective.

3. It is often alleged that schools cannot afford any incremental investments over teacher salaries. This is demonstrably untrue. What is true is that capital investments by schools are somewhat unpredictable and generally driven by broad social forces. However, the initial investment of schools in primitive computers, the massive broadening of school-provided transportation, the introduction and technological modifications of school lunch facilities, and the massive investments in asbestos removal (often uncorrelated with risk) show that schools can make major investments. What needs to be understood is what it takes to trigger such an investment.

4. It is important to note that the "cost" of software is best seen in terms of the organizational barriers to its use. For example, one product might cost $10,000 while another might cost $1,000 but require a $1,000 modification to available hardware platforms. If it is easier for a decision maker to get $8,000 more than to work around organizational restrictions on hardware purchases or modifications, then to that decision maker, the $10,000 product is more feasible. In absolute

terms, software will be seen as feasible if the effects it can produce outweigh the organizational costs, financial and otherwise, of putting it into use.

5. The Air Force uses *test stations* to diagnose and repair components from aircraft. For example, when a navigation component malfunctions, it is replaced with a unit known to be working and then is sent back to a repair shop for diagnosis and repair. In that shop, a test station is used to facilitate the diagnosis. The test station is like a giant telephone switchboard, connecting the aircraft module being tested to both power sources and measurement devices. Automated test stations use computer programs to carry out a sequence of tests to localize aircraft module faults, while manual test stations rely on a technician setting switches in response to a printed protocol in the Technical Orders for the station. Sherlock is targeted at a manual test station job. The really hard part of the job, which is what Sherlock coaches, arises when the test station itself is not working right. Then, diagnosis must proceed without a protocol that has been totally prespecified, and meters must be attached by hand to various test points on the station. Test station failures often take eight to twelve hours of diagnosis before they are found and remediated. Much of this time is spent physically reaching various test points and waiting for spare parts to arrive from a central depot. Sherlock compresses test station diagnosis to about a half hour of concentrated cognitive activity per fault isolation problem.

6. In our own work, about 20-25% of trainees are female. We occasionally use masculine pronouns in this paper, purely to simplify exposition and not to suggest that our results are gender specific.

7. The pronoun *we* is convenient but inaccurate. The Air Force Human Resources Laboratory carried out the evaluation study with our support. Official reporting of their results will appear in Nichols et al. (forthcoming). The summary we provide is not endorsed by the Air Force. While we have attempted to accurately convey our best understanding of the results, the official Air Force position on the methods employed and results obtained may deviate from our interpretation. We did conduct additional evaluations on our own, which are

reported in Lajoie and Lesgold (1990). Those results are consistent with the present discussion as well.

8. Since we use fuzzy variables rather than scalars in our current implementations, the summarizing is not done by standard regression, but conceptually, regression analysis is a good way to think about this aspect of our approach.

9. Objects should also know how to coach to maximize transfer, by emphasizing the generality of general procedures and giving situational specifics for more specific bits of knowledge, but that is not the direct focus of this chapter.

Tools for Scientific Visualization

Wallace Feurzeig
BBN Laboratories
Cambridge, MA 02138

Telephone: (617) 873-3448

1. Introduction

This paper presents an overview and assessment of computer modeling and visualization technology in science research and science education. We distinguish two distinctly different kinds of animated visualizations, *product visualization* (visualization of the model output data) and *process visualization* (visualization of the model processes *per se*). In current science modeling work at the national supercomputer centers, product visualization is extensively used but process visualization is virtually absent. Our thesis is that both kinds of visualization are valuable for gaining scientific insight and understanding. Further, we feel that their integrated use will become essential as models become more complex.[1]

Visualization is valuable in education as well as research. To make computer modeling methods accessible to students, however, we must provide semantically transparent visual representations of both model structure and model behavior. To support the conceptual clarity required for student modeling work we have developed new computer tools for visualization of model processes. Although designed for education, these tools provide some capabilities that are more advanced than those available to researchers. We will describe two such tools and suggest that the use of such tools would also have significant benefits for science research.

2. Computer Modeling and Visualization in Science Research

Computers are beginning to transform the way science is done. Scientists are using computers to model complex processes of diverse phenomena, ranging in scale from the inner structure of

---

[1] Some models are becoming so comprehensive that no scientist is expert on all aspects (e.g., global warming supermodels integrating diverse interacting submodels each of which describes phenomena such as the air-ocean interface,

classical paradigms of experiment and theory. A computer model is both the concrete embodiment of a theory and a new kind of laboratory for exploration and experiment. Computer modeling can be an illuminating source of creative insights about the structure and behavior of complex phenomena that were previously inaccessible, and it has made possible the solution of problems previously thought unsolvable. Further, modeling provides a powerful bridge between theory and experiment, informed by each to guide the other in a new synergy that extends and enhances scientific inquiry. It has already made invaluable contributions to frontier research in astronomy, biology, chemistry, physics and meteorology.

These developments have been greatly accelerated by the use of supercomputers coupled with powerful graphics display processors. Indeed, in many cases they would not have been possible without the use of such resources. When the phenomena being modeled have high-dimensional nonlinear interactions, traditional (numerical or graphical) presentations of results are not readily informative. As supercomputers are used with larger and more complex models, visual presentations become essential for understanding the results of modeling runs. In current practice at supercomputer centers the end result of "scientific visualization" is to turn the numbers generated by modeling runs (the model output data) into pictures, typically in the form of computer movies. These reveal far more vividly than can numbers, the behavior of the phenomena being modeled and the effects of model processes and interactions.[2] Portraying numerical results as three-dimensional images moving through time with color encoding produces a visually compelling and highly informative presentation that greatly aids comprehension and interpretation of model output data.

There are extensive applications of computer visualization methods of this kind in current science research (McCormick, DeFanti, and Brown, 1987; Cromie, 1988; Cassidy, 1990; Haber, 1990). A recent issue of *the International Journal of Supercomputer Applications*, (Follin, 1990) for example, includes articles describing modeling applications in climatology, planetary studies, fluid dynamics, automotive engineering, and

---

[2] "The most exciting potential of the wide-spread availability of visualization tools is not the entrancing movies produced, but the insight gained and the mistakes understood by spotting visual anomalies..." (McCormick et al, *1987*).

Effects of Increased Greenhouse Gases on Global Climate

Evolution of Severe Thunderstorms

Oceanography of the Pacific Ocean

Spectral Classification of Jupiter's Clouds

Visualization of Flow in Computational Fluid Dynamics

Three-dimensional Viscous Flow in Gas Turbines

Flow through Biofluid Devices (Artificial Heart)

Automobile Side Member Collapse

Visual Simulation of a Chemical Reaction

Glass Structures and Transition

Quantum Chemical Molecular Models

The papers include snapshots of the visual outputs of the models. An accompanying videotape shows the animations of the output data generated by each of the models. Some, like the animation of a developing thunderstorm, are vividly realistic and lifelike; some show real objects that would otherwise be unseeable.

The complex models that are run on supercomputers are usually developed by researchers on a workstation, prior to high-speed "production runs" on the supercomputer facility. The key roles of the supercomputer facilities in current practice are: supporting the execution of computation- and data-intensive models, and producing appropriate numerical data for subsequent analysis. The post-processing of the results of model computations to turn the numerical outputs into animated graphical presentations - pictures or movies - is what is meant by "scientific visualization". This phase often involves the use of powerful graphics workstations.

The representation of a scientific model as a computer program is a complex process involving representations at several levels of abstraction: 1) the conceptual entities that the scientist envisages in his or her mental representation (i.e. the objects and processes being modeled), 2) a mathematical description of the behavior and interactions of these objects, 3) a computer program for implementing the mathematics, and 4) the visual representation of the results obtained from running the model. Typically, in current scientific practice, the conceptual entities are described by differential equations, Fortran is used for transforming the equations into programs, and Wavefront or a similar type of graphical rendering and presentation system is used for transforming the outputs of Fortran programs into scientific visualization, correspond more directly to the conceptual entities in the scientist's mental model than do the intermediate representations.

3. Computer Modeling and Visualization in Science Education

Under an NSF project supported by the Applications of Advanced Technologies Program of the Education and Human Resources Directorate[3], we are exploring the educational applications of new computer modeling tools and paradigms. Our interest is motivated by several considerations. The arrival of affordable personal computers with the computational power of present-day supercomputers is imminent.[4] Hardware systems with the capabilities of today's supercomputers will be arriving in schools during the next decade. We must start to think now about how they should be used. We need to develop the appropriate ideas, software tools, learning activities, and exemplary demonstrations.

The quantitative improvements in performance embodied in these new machines make possible *qualitative* changes in the nature of computing. These changes can be exploited to provide enormous educational benefits. In particular, real-time interactive models with richly animated graphics displays, the same kinds of tools being used with great benefit in science research, can be made accessible for use by students. The models and the modeling tools students work with will be a great deal simpler than those used by scientists, but the fundamental character of the modeling activity will be the same, as it should be. The current school science course focuses on teaching *about* science. Instead, students should be *doing* science. The way is open to introduce modeling into schools as a compelling new paradigm.

Computer modeling is valuable for students for very much the

---

[3] NSF Grant MDR-8954751, "Visual Modeling: A New Experimental Science".

[4] *The 80860, a million-transistor chip that will be available this year, is "designed to be a microprocessor version of a Cray supercomputer."* (Byte, April, 1989). Like the Cray, it has 64-bit data paths and a pipelined architecture that can perform multiple floating-point operations in parallel (120 million such operations per second). It has a graphics coprocessor that produces three-dimensional color shadings. It will sell initially for just $750, and that price is certain to drop substantially over the next few years. By the early 1990's it will be possible to design a personal computer with the power of a mainframe of today, using fewer total chips than those in a Mac II.

phenomena that are not accessible to direct observation, thereby enhancing their comprehension of underlying mechanisms. It can provide insight into the inner workings of a process or phenomenon - not just about *what* happens, but *why* it happens. It enables one to make and test predictions, and to ask and answer questions such as "What will happen if this parameter is changed?", "What are the critical dependencies?", and "How can one modify or extend the model so as to produce a specified behavior?" It enables investigation in situations where experimentation may be impractical or infeasible, and it enables the modeler to gain more information about a process than can be obtained otherwise, e.g., by slowing down or speeding up time or by presenting simultaneous multi-window views of different representations.

Computer modeling can dramatically enliven science education. It has unique capabilities for providing students compelling experiences, engaging them in active investigation, and enhancing their scientific understanding. A curriculum centered on modeling activities can foster the development of the notions and art of scientific exploration and inquiry. Modeling microworlds can incorporate powerful graphic interfaces to enable easy interaction without the need for a deep understanding of computers. It can support facilities that demonstrate concepts and that aid students in solving problems. A computer modeling approach to teaching science has the potential for motivating the interest of significantly greater numbers of students, not just the small fraction who are already turned on to science and mathematics.

Computer modeling is not new. Modeling languages and applications have been in use in education for some time.[5] What is new is the possibility of making complexity more comprehensible and accessible to students through the use of new kinds of modeling tools made possible by the more powerful computational systems that are on the way. Most people have difficulty understanding the dynamic behavior of systems composed of interacting subsystems. Modeling tools that do not support process visualization have proved ineffective in helping students gain insight into the mechanisms underlying the behavior of complex systems.

These difficulties would surely be exacerbated by the

[5] See, for example, (Roberts and Barkley, 1988; Tinker, 1990).

processes are ubiquitous. They are fundamental in nature, in the phenomena of physics, chemistry, biology, and psychology at all levels. They are essential components of complex interactions in everything that's interesting to us. Real objects in the world function, dependently and interdependently, "at the same time." In chemical reactions different processes occur simultaneously, and success depends critically on timing. Biological models of growth and change require that different processes occur continuously and synergetically. Adaptive systems – the brains of animals, ecological systems, and social organizations – are intrinsically parallel. We need to make the underlying principles more transparent and comprehensible. We need visual modeling paradigms with better visual representations for thinking about parallel processes and complex systems.

Notwithstanding its great utility our experience is that visualization of model *outputs* (product visualization) does not go far enough in promoting students' understanding of "why things turned out the way they did". Even when the output of a model run gives a complete picture of the behavior of the modeled system, an understanding of how the system operated to give rise to the results may not be evident. Moreover, the behavior that is depicted may be partial and incomplete in important aspects, even when it looks right. This is analogous to a classic situation in microbiology specimen analysis: what is salient in the microscope display of a sample visually enhanced by staining may depend critically upon the particular staining reagent used. Another reagent may reveal significantly different and informative features. The microbiologist who fails to take that into account may make incomplete, and even faulty, inferences about structure and underlying mechanism.

There is another, more insidious, problem with product visualization. Visualization techniques such as three-dimensional rendering using shading and color, together with smooth animation, can produce an illusory world that is visually compelling and can seem so real that it threatens to overwhelm and hide the simplifications and defects inherent in any computer-based model of the real world. The results of even quite simplistic models can take on a superficial credibility, reflective more of the sophistication and attractiveness of the display technique than of the model itself. This is especially a concern for education. Students may well be led astray by faulty visualizations, particularly when these are coherent and seem beautiful. (This is problem in the context of work with supermodels, in applications integrating several complex models where there are strong interactions among the constitutent models and where no scientist is an expert in all the areas modeled.)

We believe that both the model development process and the analysis and interpretation of model run data can be greatly enhanced by the introduction of new tools expressly designed to complement product visualization. We think these kinds of tools will be useful to professional scientists as well as students. There is need, first, for a new kind of visual modeling facility, a "process visualization" tool for animated graphic presentation of the *model processes themselves*, the submodel structures and algorithms, as they interact during the run. One might call this "front-end" visualization. to distinguish it from the "rear-end" visualization of the model's outputs that is an established part of current supercomputer practice (i.e., product visualization). The new tool is intended to provide a visual isomorph of the student's (or scientist's) conceptualization of the model, so as to show the objects being modeled and their interactions in as direct and transparent a way as possible. We describe two kinds of process visualization tools in the sections following.

4. Function Machines, a Visual Programming Language

One of the process visualization tools we have developed is a visual programming language called *Function Machines*. *Function Machines* uses two-dimensional iconic representations of programs, in contrast with the familiar (one-dimensional) textual languages in almost universal use today. Our primary objective in developing *Function Machines* was to make programming easier to use for mathematical exploration and inquiry. In working with educational programming languages, even with the most accessible languages like Logo, students and teachers often have difficulty understanding control structures and acquiring fluency using iteration and recursion, which are central for the description of algorithms. These conceptual barriers to acquiring a non-superficial level of programming competence have largely been eliminated in *Function Machines*.

In *Function Machines* the central metaphor is that a function (or procedure or algorithm) is a "machine" (displayed as a rectangular icon with inputs and outputs). A machines's data and control outputs can be passed as inputs to other machines through explicitly drawn connecting paths. Any collection of connected of arbitrary complexity level can be constructed. As machines are activated and run, their icons are shown in inverse video, and the passage of data into and out of machines is shown by animating the data and control paths. Thus, the operation of a *Function Machines* program is visually explicit and very easy to follow. A brief look at *Function Machines* follows, to show the visual representation of algorithmic processes within this paradigm, and its use in mathematical modeling. The *Function Machines* language and *Function Machines* programming are described more fully in (Wight et al, 1988)

The left side of Figure 1 shows a machine that computes the logistic function, $f(t)=\mu t(1-t)$. The machine contains two input "hoppers" (one for the parameter, μ, and the other for the argument, t , as the labels under the corresponding hoppers indicate) and one output "spout" that will contain the result of the calculation. The right side of the figure shows the internal structure of the Logistic machine. The inside of the Logistic machine contains three simpler machines:[6] two multiply machines, denoted by "*", and a subtraction machine, denoted by "—". When numerical values for μ and t are supplied to the Logistic machine's hoppers, it passes them to its internal machines, which perform their indicated functions to carry out the computation. Data are moved from hoppers to machines and from one machine to another along the connecting lines shown (called "pipes"). The result of the calculation is sent to the Logistic machine's output spout.

[INSERT FIGURE 1 ABOUT HERE]

As shown in Figure 2, the process of building more complex machines out of simpler ones can be continued to higher levels of embedding. The upper left window of the figure shows a composite machine, the Parallel Logistic, which is composed of composite machines. It has a single input, μ,(whose current value is 3.8) and it has no outputs. The inside of this machine is shown in the right window. It is seen to be composed of three composites, two Logistic machines and a Scatter Plot machine. The two Logistic

[6] These are "primitive" machines provided by the system as building blocks for constructing more complex machines (programs). There are more than 100 such primitives including the mathematical and logical functions and the graphics and input/output constructs typically found in programming languages.

machine and the other feeds back as the next input value of t (this is a straightforward visual way of directing the simplest form of iteration, *backput iteration*, where each output becomes the next input). The inside of one of the Logistic machines is shown in the lower left window of the figure. The Scatter Plot machine produces an x,y point plot from its two inputs (i.e., a scatter plot graph). Its inner structure is not shown.

[INSERT FIGURE 2 ABOUT HERE]

Figures 3 and 4 show the results obtained from running the Parallel Logistic program. Initially the output values of the two Logistic machines (whose initial inputs differed by only 0.000001) are very close together, so the points generated by their scatter plot lie on a diagonal as shown in Figure 3. Subsequently, however, the two Logistic outputs become widely divergent as is shown in Figure 4. (This illustrates a characteristic behavior of the phenomenon known as mathematical chaos - the great sensitivity of nonlinear functions, even simple quadratic functions like the Logistic, to small differences in initial conditions).

[INSERT FIGURES 3 AND 4 ABOUT HERE]

The logistic program shows the use of *Function Machines* for process visualization of mathematical models defined by equations. *Function Machines* can also be used for process visualization (and product visualization) of one class of models defined by *object-oriented simulations*. The *Function Machines* visual programming language supports a single class of objects, *graphic turtles*. Like objects in general, turtles have state variables (including their location and heading). They also have algorithms (called "methods" in object-oriented programming jargon) for such actions as moving forward a specified distance, turning a specified angle, and drawing their icon to show their current location and heading.

Figure 5 shows the *Function Machines* program and initial display for modeling an interactive multi-turtle simulation called Turtle Tag. The classic turtle tag problem is to describe the pattern generated by the tracks of four turtles, initially postitioned at the vertices of a square, who simultaneously move in a counterclockwise direction toward their nearest neighbors. The turtle display (in the right window) shows the initial positions of the turtles. As the program runs, each turtle first computes the heading of its nearest neighbor. Thus, turtle *a* process continues with further rounds of seeks and moves.

The left window of Figure 5 shows the top-level program. First, the four turtles are created and given their initial locations and headings (by the Create Turtles machine). Next, the four Seek machines compute the new headings for turtles *a, b, c,* and *d,* respectively. Then, the four Move machines move the turtles forward a fixed dis' ³ ıce along the new headings. The output of each Move machine passes the current position and heading of its turtle to the appropriate Seek machine to ready it for its next computation.

[INSERT FIGURE 5 ABOUT HERE]

The right window of Figure 6 shows the inside of one of the Seek machines, that for turtle *b* seeking turtle *a*. The Seek machine contains two primitive turtle machines, Get XY, which computes and outputs the x and y location of turtle *a*, and the Head Towards machine, which has three inputs: turtle *a*'s x and y coordinates, and the name of the turtle that is to move to that location (in this case, turtle *b*). The other three Seek machines effect the same actions for turtles *c*, *d*, and *a*, respectively.

[INSERT FIGURE 6 ABOUT HERE]

The inner constituents of the Move machine are not shown here. (It contains two primitive turtle machines, Set Heading, which sets the heading computed by Seek, and Forward, which moves the turtle a designated distance toward the computed heading).

Figure 7 shows the Turtle Tag program in operation. As the left window shows, the four Seek machines are ready to run. Note that all four have been activated at the same time so they will run concurrently. The program has been in operation for some time. The right window shows the tracks that have been generated by the turtles thus far.

[INSERT FIGURE 7 ABOUT HERE]

Figure 8 shows the program at a later time. The left window shows the four Move machines being invoked concurrently. The right window shows the spiral tracks that have been generated by the turtles. At this point their paths have converged - the four turtles are virtually colocated.

Figures 7 and 8 demonstrate the simultaneous presentation of process and product visualizations. As the program runs one can see the processes that are currently computing. At the same time one can also see what effects these processes have on the model's visual outputs. Moreover, one can study the relation between the program description and the program output more intensively by running the program incrementally at one's own pace, one step at a time. Observing the model processes in animation can give students very direct insight into the mechanisms underlying the model outputs. The educational benefits from working with both kinds of visualizations increase as models become more complex.

5. Cardio: Object-Oriented Simulation Modeling

Turtle Tag is an example of a relatively simple model with concurrent processes. The processes in Turtle Tag are both autonomous and synchronous. Many phenomena of interest in science are a great deal more complex, often involving real-time concurrent processes with time delays, feedback loops, and asynchronously coupled constituents. To model such phenomena in a way that adequately captures and expresses these complex behaviors imposes computationally intensive requirements that tax the capabilities of most visual programming languages and visual modeling systems.

An example of a model that addresses all these characteristics of complex concurrent phenomena is Cardio[7], an object-based modeling system expressly developed to model the processes that underly the dynamics of the heart's electrical control system. Cardio provides students an interactive visual simulation environment for investigating the physiological behavior of the human heart while gaining insight into the dynamics of oscillatory processes in general (particularly coupled oscillators, which are fundamental to the operation of living systems). Cardio generates process and product visualizations of the heart's pattern-producing electrical control system. It enables students to investigate the deterministic heart dynamics produced by the cardiac electrical system and to study the effects of changes to specific heart component parameters.

[7] The Cardio program was designed and implemented by Eric Neumann of BBN under NSF Grant MDR-8954751, "Visual Modeling: A New Experimental Science". Cardio is based in part on descriptions of heart dynamics in (Bratko et al, 1989; Glass and Mackey, 1988; and Winfree, 1987).

The major product visualization is the *animated heart display*, shown in Figure 9. This is a system-driven graphic animation of the heart model which shows in real time the rhythmic pulsation of the heart chamber accompanied by the sound of the opening and closing of the heart valves. Subsequent figures show the animated heart in other phases of its operation to give a sense of its dynamics.

[INSERT FIGURE 9 ABOUT HERE]

There are three main types of tissue components in the heart model: the two pacemakers (the SA node and the AV node) which are pulse generators with a natural frequency, but they are also sensitive to external signals which can reset them; the conduction paths (which have associated time delays); and the heart chamber muscles (the two atria and the two ventricles), which are the target tissues of the electrical action. All three kinds of tissues are excitable media. All have refractory periods for recovery after triggering. These, together with the time delays, are the sources of the complex nonlinear behavior of the system. The components are all represented as objects in the model (in the sense of object-oriented programming constructs). Their operation is shown schematically in an animated display, the *electrical control schematic*, which is the major process visualization of the model. The electrical control schematic is shown in the left window of Figure 10, which also shows the animated heart display.

[INSERT FIGURE 10 ABOUT HERE]

The SA pacemaker node is shown at the top of the electrical control schematic display. It has conduction paths to the two atria, and to the AV pacemaker node which, in turn, activates the two ventricles (note that it first goes to an intervening node, which has a single path to the leftmost ventricle and a dual path to the rightmost one). When the model runs, the moving electrical action potential is shown with animated trigger pulses, as seen in Figure 11 which shows the pulsing from the AV pacemaker along the common conduction path leading to the ventricles.

[INSERT FIGURE 11 ABOUT HERE]

The electrical control schematic is also the user's control panel. The icons shown at the left column of the schematic display in Figures 10 and 11 represent user-selectable tools for setting model parameters, and other model control functions.

The heart system dynamics result from the run-time interactions of the pacemaker nodes, conduction paths, and heart chamber muscles. The interactive heart animation and electrical control schematic displays can be simultaneously viewed with EKGs and phase plots showing heart dynamics. EKG graphs are constructed and displayed in real time from the 3-dimensional dipole field generated by the four chambers. The depolarization wave of the myocardium creates a positive deflection on the EKG trace as the wave approaches a lead, a negative deflection as the wave recedes, and no deflection if the wave moves orthogonally to the lead. The leads represent the difference between each pair of contact positions (i.e., right arm, left arm, and feet). Based on the interpretation of at least three different EKG leads, sophisticated users are able to reconstruct the 3-dimensional electric vector time-dependent sweep of the heart. Conduction delays between the atria and ventricles will appear on the tracings as delays between the deflections. The top right window of Figure 12 shows a typical EKG plot displayed together with the electrical control schematic.

[INSERT FIGURE 12 ABOUT HERE]

EKGs are useful in identifying pacemaker characteristics, conduction rate changes and myocardium anomalies (e.g., ischemia and infarcts). However, because EKGs are the result of the combined electric fields of each chamber, it is not easy to visualize from EKG plots alone, the complex and asynchronous patterns of chamber depolarization that continuously evolve over time. Such patterns may arise when the heart does not return to the same state-space after a single pacemaker cycle (as is the case for myocardium which is still in the refractory state caused by the previous pulse). To help visualize such complex behavior, phase plots of the contractions or electric fields of one chamber plotted against those of another chamber illustrate the dynamics by means of orbit paths. A phase plot of the right atrium contraction vs. left ventricle contraction is shown in the bottom right window of Figure 12. The plot shows a limit cycle; its eccentricity depends on the phase difference between the two chambers.

Complex dysrhythmias can be generated and their effects on heart dynamics observed in the animated heart display and analyzed not always follow the sinus pacemaker, producing multiple orbit paths like those shown in the phase plot in Figure 13.

[INSERT FIGURE 13 ABOUT HERE]

During a simulation, the student is able to record and plot various time-dependent dynamic variables, including EKGs and phase plots of chamber contraction. This is useful in comparing the dynamics of systems with different parameter values. The user is able to inspect and modify the parameter settings for each component of the model. The bottom right window in Figure 14 shows a display (called by the user) of the SA pacemaker node. The display gives information about the function of the node and shows the current values of the key parameters associated with it. The user can modify these values and run the simulation to see the effect of these changes on the operation of the model.

[INSERT FIGURE 14 ABOUT HERE]

The model includes sets of component parameter values for several pre-defined heart dysfunctions and dysrhythmias to enable students to investigate the dynamics of typical heart anomalies and diseases. Other types of dynamical behaviors can also be created and studied in Cardio. Mechanical, electrical, and chemical disturbances of many kinds can be introduced and their effects on heart behavior observed and analyzed. Cardio-pharmacological agents such as digitalis and adrenergic compounds are being implemented so that their physiological effects on component parameters can be studied. Multiple (i.e., ectopic) pacemakers can be modeled in several ways (e.g., resetting and non-resetting). Combined with the intrinsic refractory limit of the conduction system, these yield complex echo and skip beats. By saving heart parameters as files, Cardio enables students to compare, model and test various heart conditions and to determine the state-space domains of complex and chaotic rhythms. We plan to incorporate machine-based laboratory (MBL) probes into Cardio to enable students to measure, and observe in real time, the performance of their own hearts.

Because the heart model derives its behavior from component interaction, students can change the parameters of any component or add their own components to form ectopic pacemakers and anomolous conduction paths. Further, Cardio's visual modeling tools enable students to graphically create new components by using new instances of pre-compiled objects and inserting them into the component list, circumventing the use of a slower and less efficient interpretive structure. Thus, students can easily and quickly create their own heart models and investigate their behaviors. For example, Figure 15 shows the electrical control schematic created by a user to investigate the dynamics of a heart with only a single pacemaker, to try to gain a very specific understanding of the advantages provided by the more complex double pacemaker human heart system.

[INSERT FIGURE 15 ABOUT HERE]

6. New Visual Modeling Tools

We have argued that the incorporation of process visualizations is a highly desirable addition to the technology of computational modeling, and an essential one both for the introduction of modeling in science education, and for the support of science research involving complex models. We have showed examples of two different kinds of process visualization tools. One kind showed the kind of visualization enabled by the use of a "universal" visual programming language capable of representing models with simple concurrent processes. The other showed the more powerful kind of visualization enabled by an object-based modeling system with a user interface expressly designed to facilitate modeling investigations, and with very specific simulation capabilities for representing some kinds of models with more complex concurrent processes.

Another new tool is needed to link the process and product visualizations by providing a visual facility for interactive program control of model inputs and outputs during a run.[8] This is, in effect, a modeler's control language. Scientists and students will want to get inside the program while it is running and feel that what's on the screen is real. They will want to be able to go in and make changes and see the feedback immediately. This kind of tool is designed to provide the user with a view of the objects and object-interactions being modeled, and to invoke the facilities for producing scientific visualizations of the

[8] "Scientists not only want to analyze data that results from modeling super-computations; they also want to interpret what is happening to the data during their computations. They want to steer calculations in close to real-time; they want to interact with their data." (McCormick et al, Op. Cit).

for input, monitoring, intervention, inspection, and display of model processes and modules (in both graphic and symbolic forms).

As we noted above, the use of compelling product visualizations does not guarantee that users will understand the model processes that give rise to them. Neither, however does the addition of product visualizations (though these clearly provide a valuable new dimension of modeling power). The possibility remains, particularly with beginning students, that modeling investigations using these visualization capabilities will be no more insightful than is often the case with the simpler modeling systems. In many such "modeling" activities, students vary parameters and generate tables and graphs, but do not gain any understanding of the underlying mechanisms relating their output data to the model's inputs and actions. The existence of attractive facilities for animating the model processes and outputs does not assure, by itself, that students (or, indeed, researchers) will gain insights or understanding. We are convinced that the incorporation of constructive facilities like those in Cardio, that enable users to construct their own models, in addition to studying and modifying given models, are essential for fostering such insight and understanding. The possibility of assigning tasks such as "build a model with the following specified behavior" is, in our view, an essential requirement for investigating complex phenomena by computational modeling.

Bratko, Ivan, Igor Mozetic, and Nada Lavrac. "*Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems*", MIT Press, Cambridge, Massachusetts, 1989.

Cassidy, Robert. "Scientific Visualization: a New Computer Research Tool", Research and Development, April, 1990, Pp, 50-60.

Cromie, William J. "Computer Images in Five Dimensions", Mosaic, Vol. 19 No. 2, National Science Foundation, Wahington D.C., 1988.

Foley, James D. "Scientific Data Visualization Software", *International Journal of Supercomputer Applications,* Vol. 4 No. 2, Summer, 1990, Pp. 154-156.

Follin, Steve, editor. "Special Video Issue", *International Journal of Supercomputer Applications,* Vol. 4 No. 2, Summer, 1990, Pp. 3-149.

Glass, Leon and Michael C. Mackey. "*From Clocks to Chaos. The Rhythm of Life*", Princeton University Press, Princeton, New Jersey, 1988.

Haber,Robert B. "Scientific Visualization: What's Beyond the Vision?", *International Journal of Supercomputer Applications,* Vol. 4 No. 2, Summer, 1990, Pp. 150-153.

McCormick, Bruce H., Thomas A. DeFanti, and Maxine D. Brown, "Visualization in Scientific Computing", *Computer Graphics,* Vol. 21 No. 6, ACM SIGGRAPH, New York, N.Y., November, 1987.

Roberts, Nancy and Tim Barclay. "Tools for Model Building and Simulation in the High School", *Journal of Computers in Mathematics and Science Teaching,* Vol. 8 No. 4, 1988.

Tinker, Robert F. "Modeling: Instructional Materials and Software for Theory Building", NSF Final Report, The Technical Education Research Centers, Inc., Cambridge, Massachusetts, 1990.

Wight, Sterling, Wallace Feurzeig, and John Richards. "Pluribus: A Visual Programming Environment for Education and Research", *Proceedings, IEEE Workshop on Languages for Automation,* College Park, Maryland, August, 1988.

Winfree, Arthur T. "*When Time Breaks Down. The Three-Dimensional*

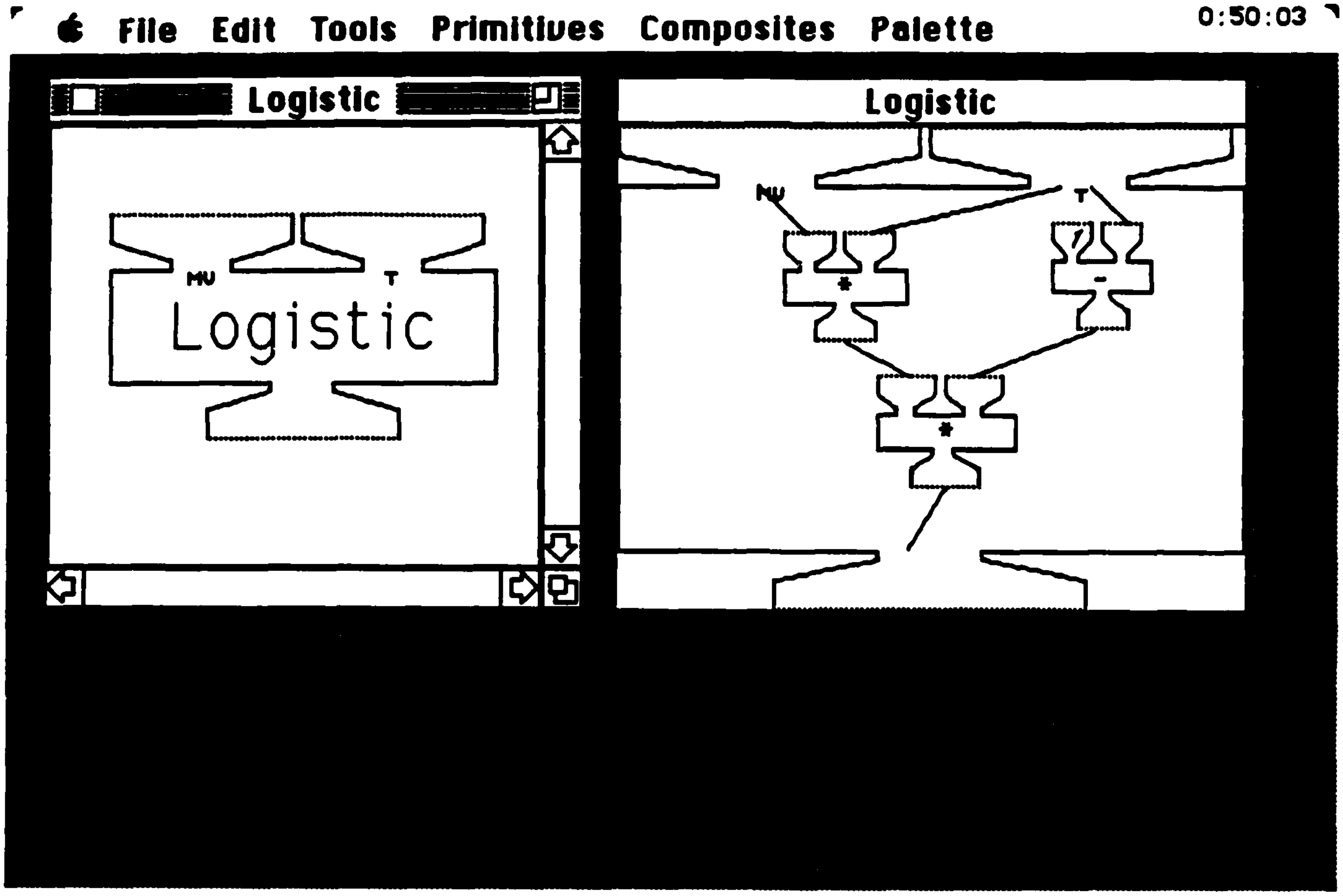

File Edit Tools Primitives Composites Palette
0:50:03
Logistic
MU
T
Logistic
Logistic
MU
T
*
/
-
*

Figure 1

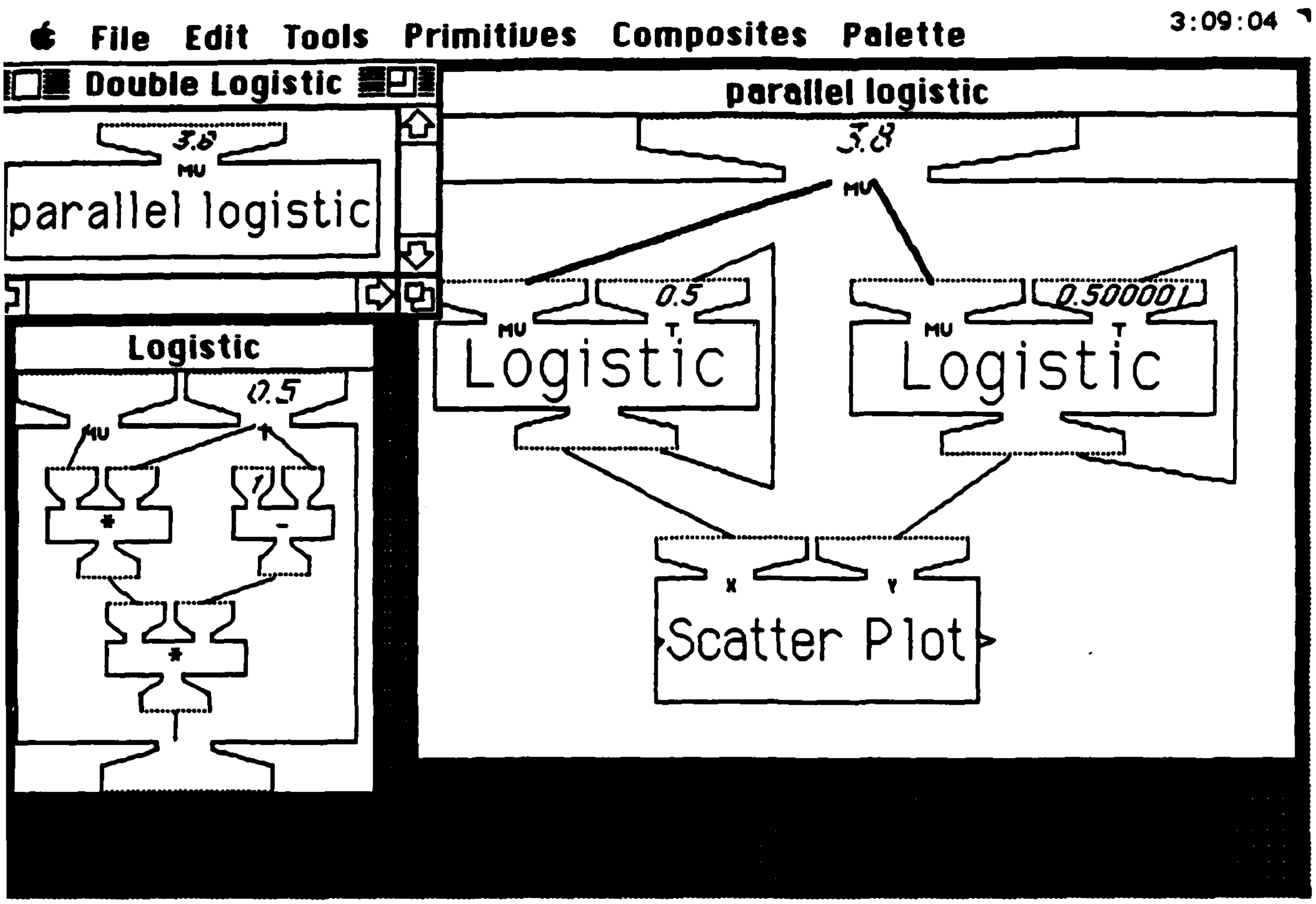

File
Edit
Tools
Primitives
Composites
Palette
3:09:04
Double Logistic
3.8
MU
parallel logistic
Logistic
0.5
MU
*
1
-
*
parallel logistic
3.8
MU
0.5
MU
T
Logistic
0.500001
MU
T
Logistic
X
Y
Scatter Plot

Figure 2

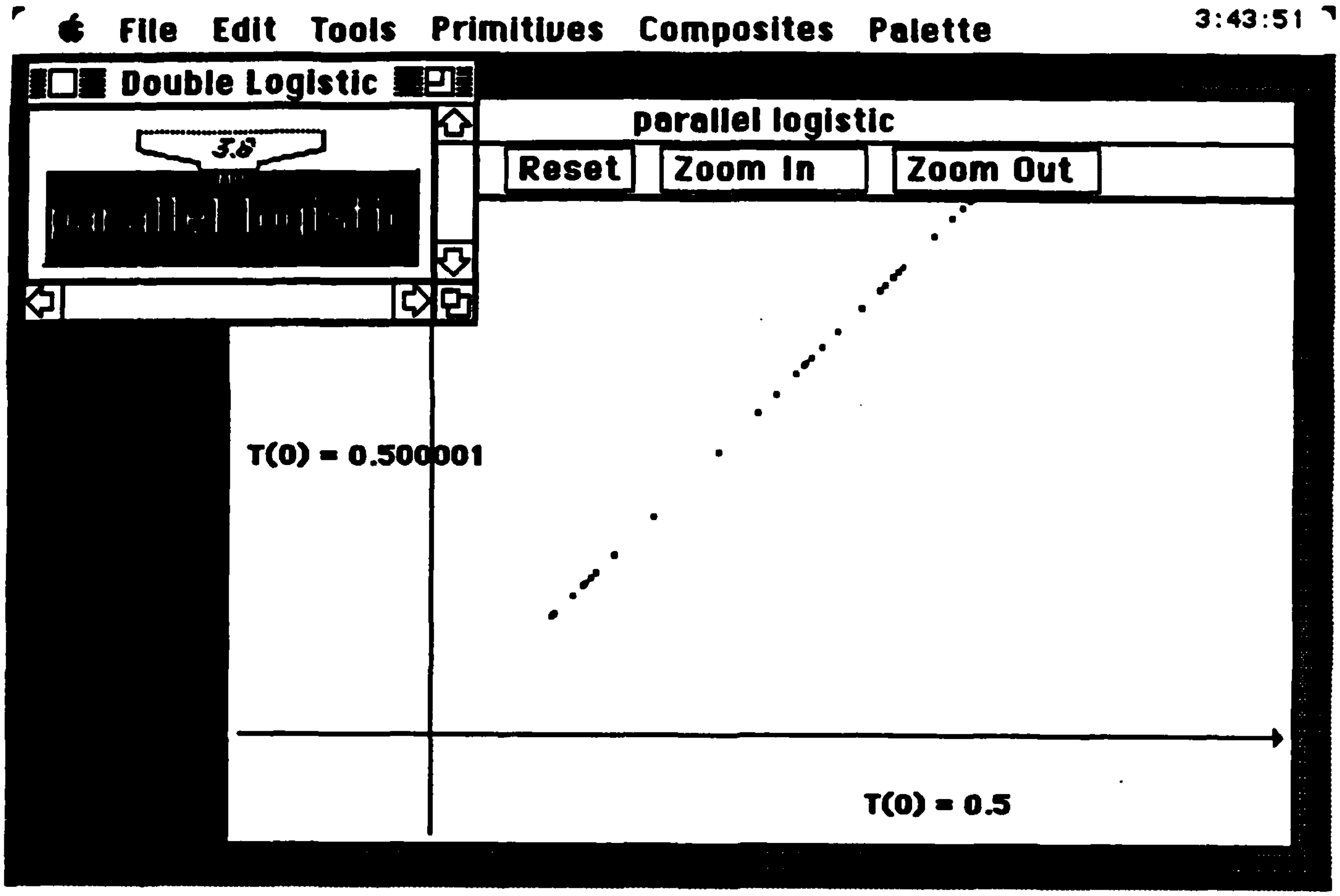

File Edit Tools Primitives Composites Palette
3:43:51
Double Logistic
3.8
parallel logistic
Reset
Zoom In
Zoom Out
T(0) = 0.500001
T(0) = 0.5

Figure 3

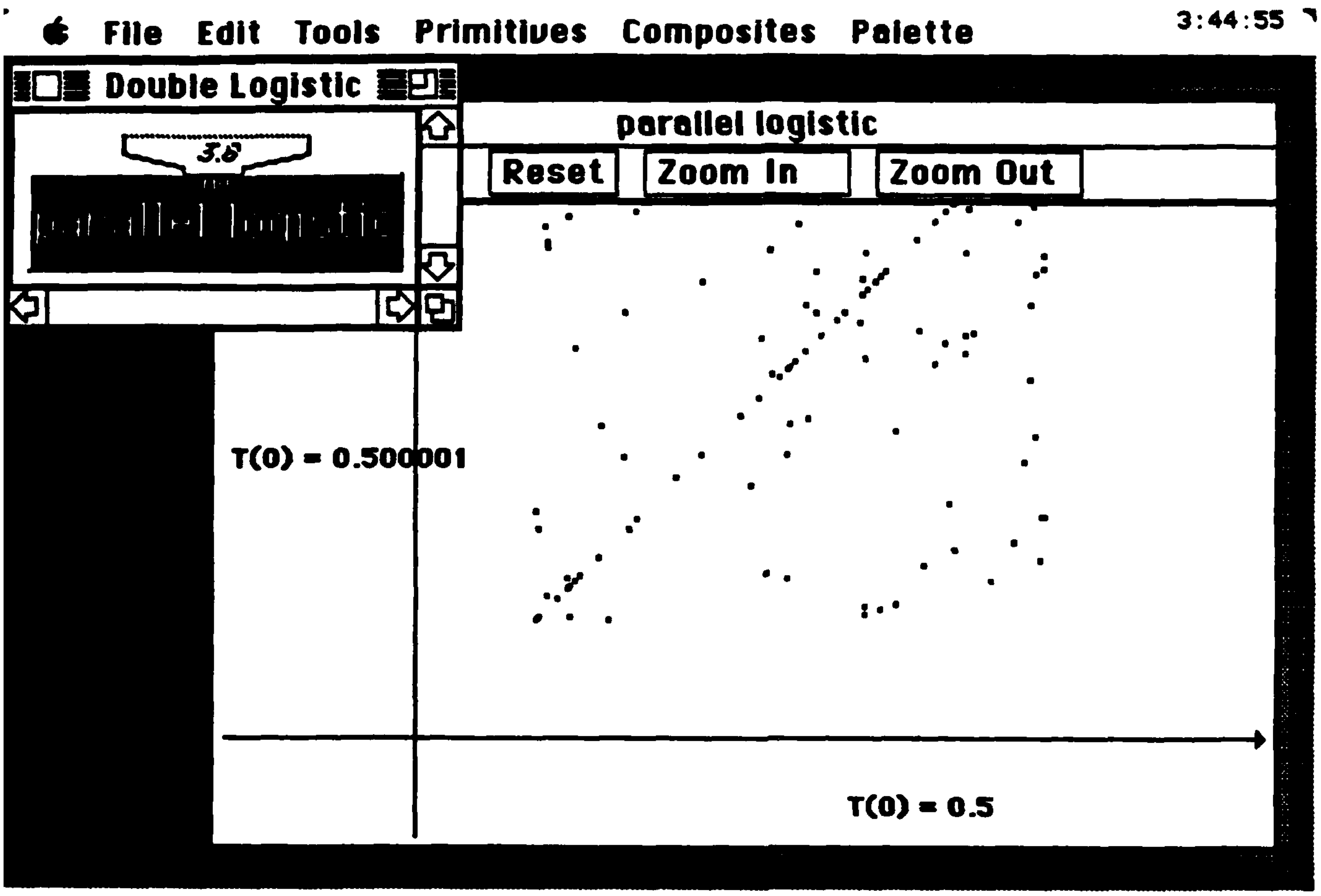

File Edit Tools Primitives Composites Palette
3:44:55
Double Logistic
3.8
parallel logistic
parallel logistic
Reset
Zoom In
Zoom Out
T(0) = 0.500001
T(0) = 0.5

Figure 4

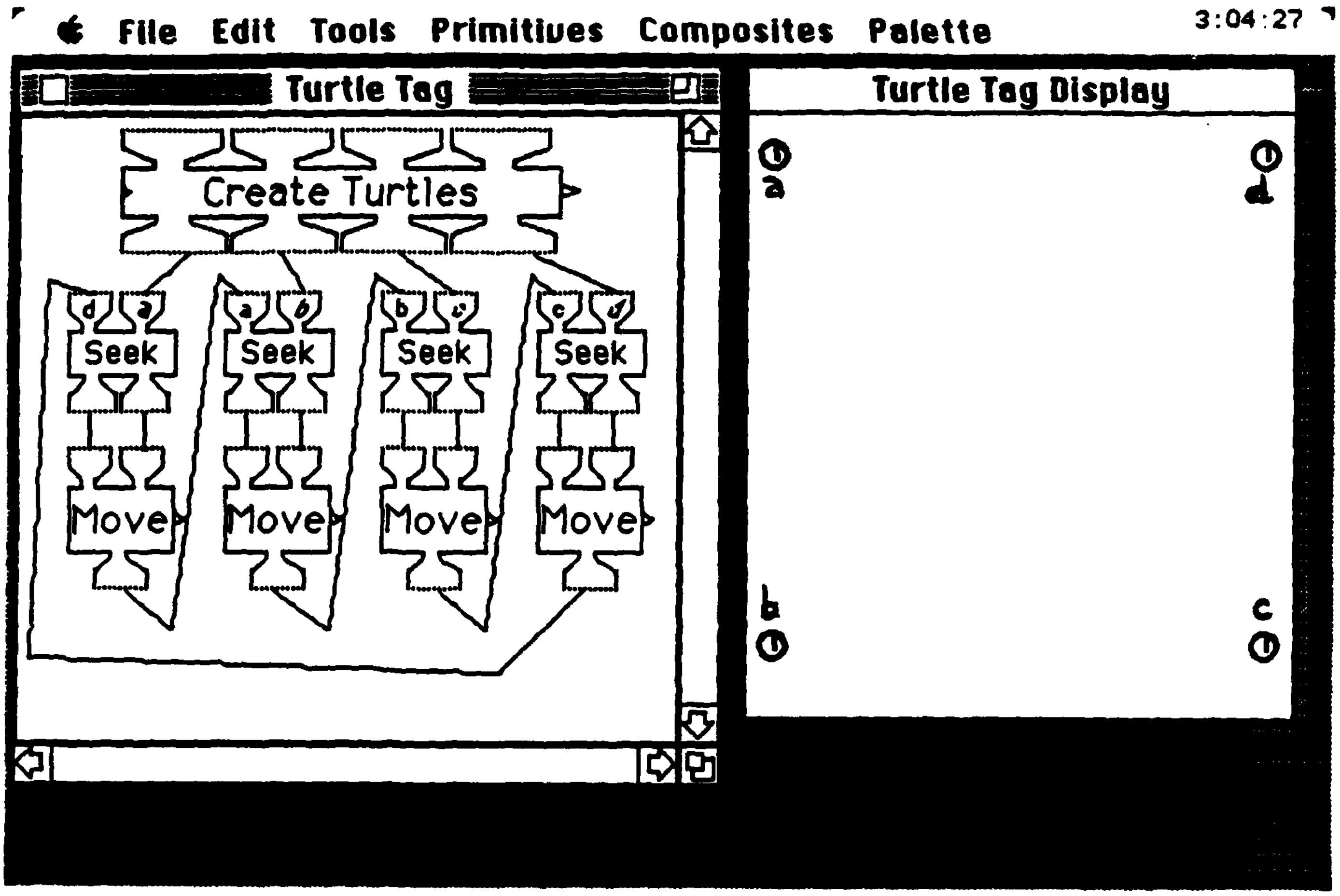

File
Edit
Tools
Primitives
Composites
Palette
3:04:27
Turtle Tag
Create Turtles
d a
Seek
Move
a b
Seek
Move
b c
Seek
Move
c d
Seek
Move
Turtle Tag Display
a
d
b
c

Figure 5

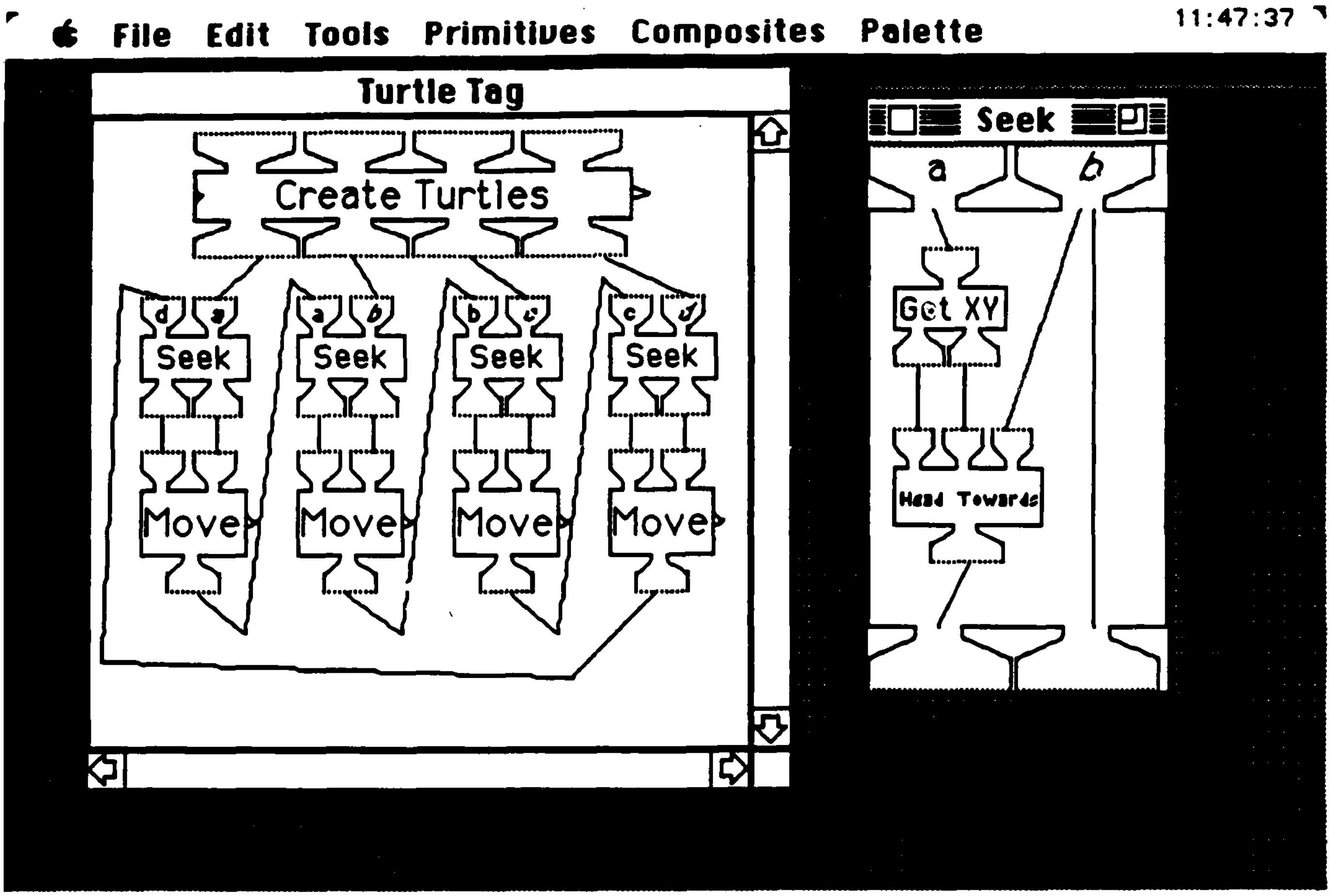

File Edit Tools Primitives Composites Palette
11:47:37
Turtle Tag
Create Turtles
d a
Seek
Move
a b
Seek
Move
b c
Seek
Move
c d
Seek
Move
Seek
a
b
Get XY
Head Towards

Figure 6

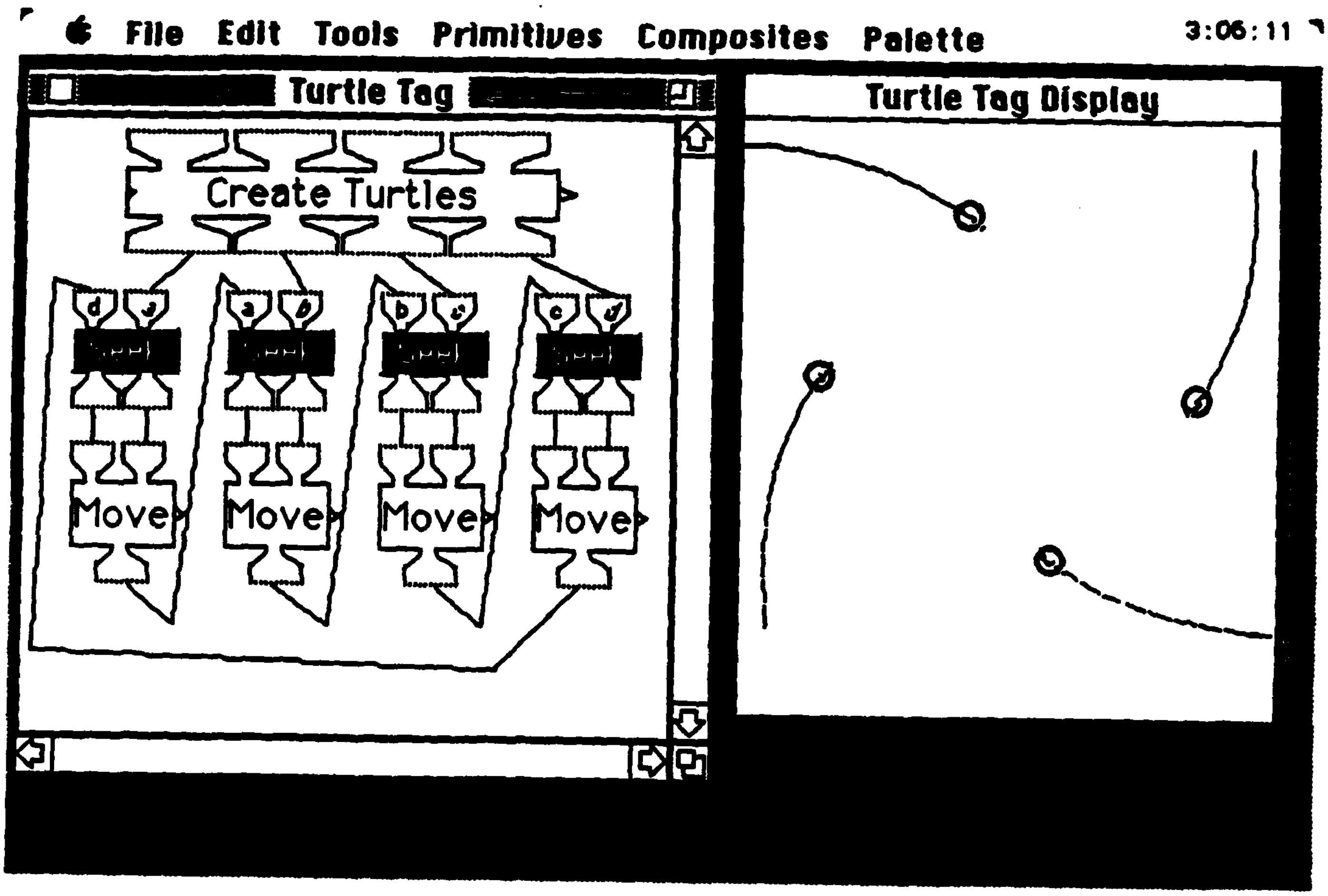

File
Edit
Tools
Primitives
Composites
Palette
3:06:11
Turtle Tag
Create Turtles
Move
Move
Move
Move
Turtle Tag Display

Figure 7

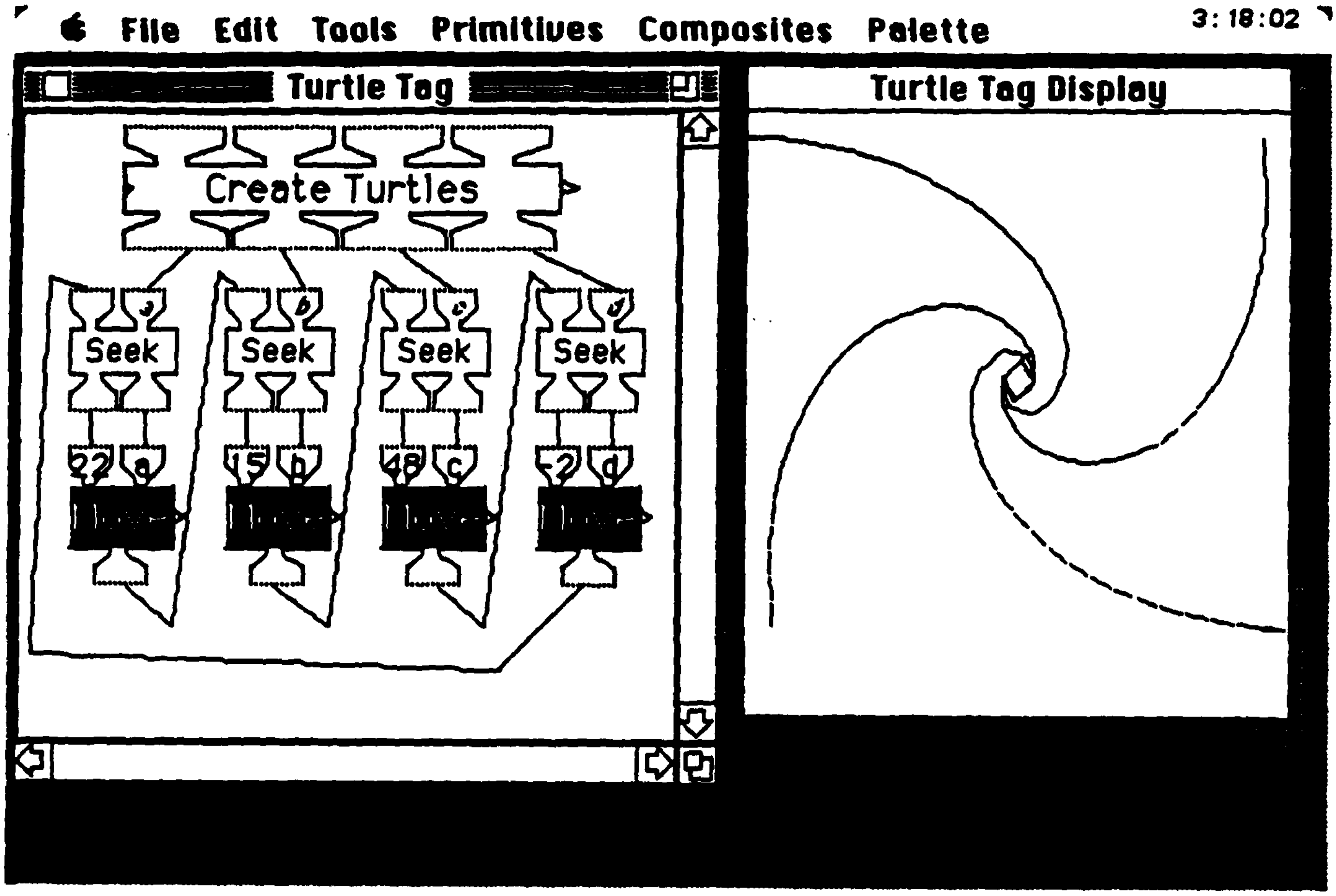

File Edit Tools Primitives Composites Palette
3:18:02
Turtle Tag
Create Turtles
Seek
Seek
Seek
Seek
22
15
48
-2
Turtle Tag Display

Figure 8

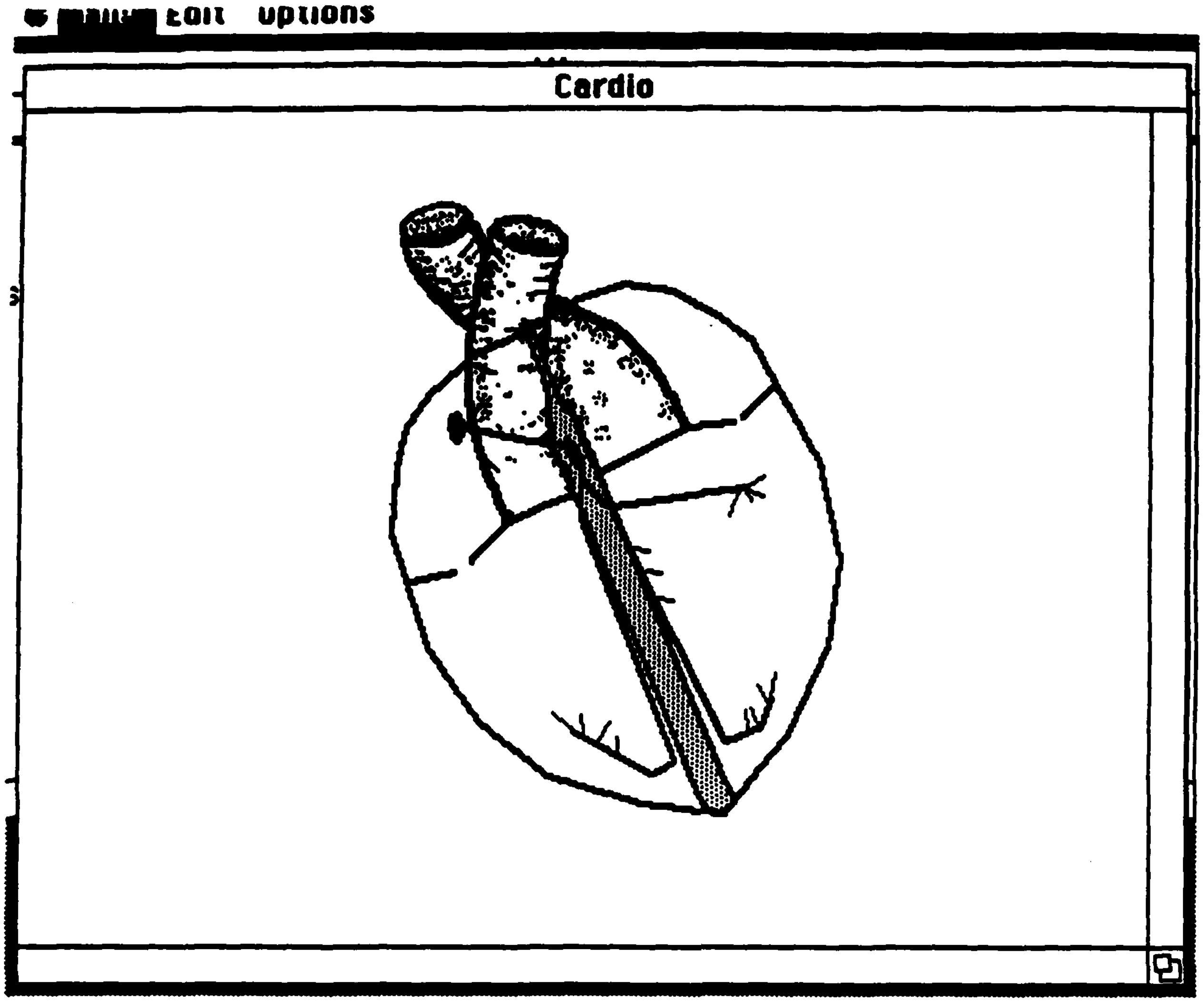


Figure 9

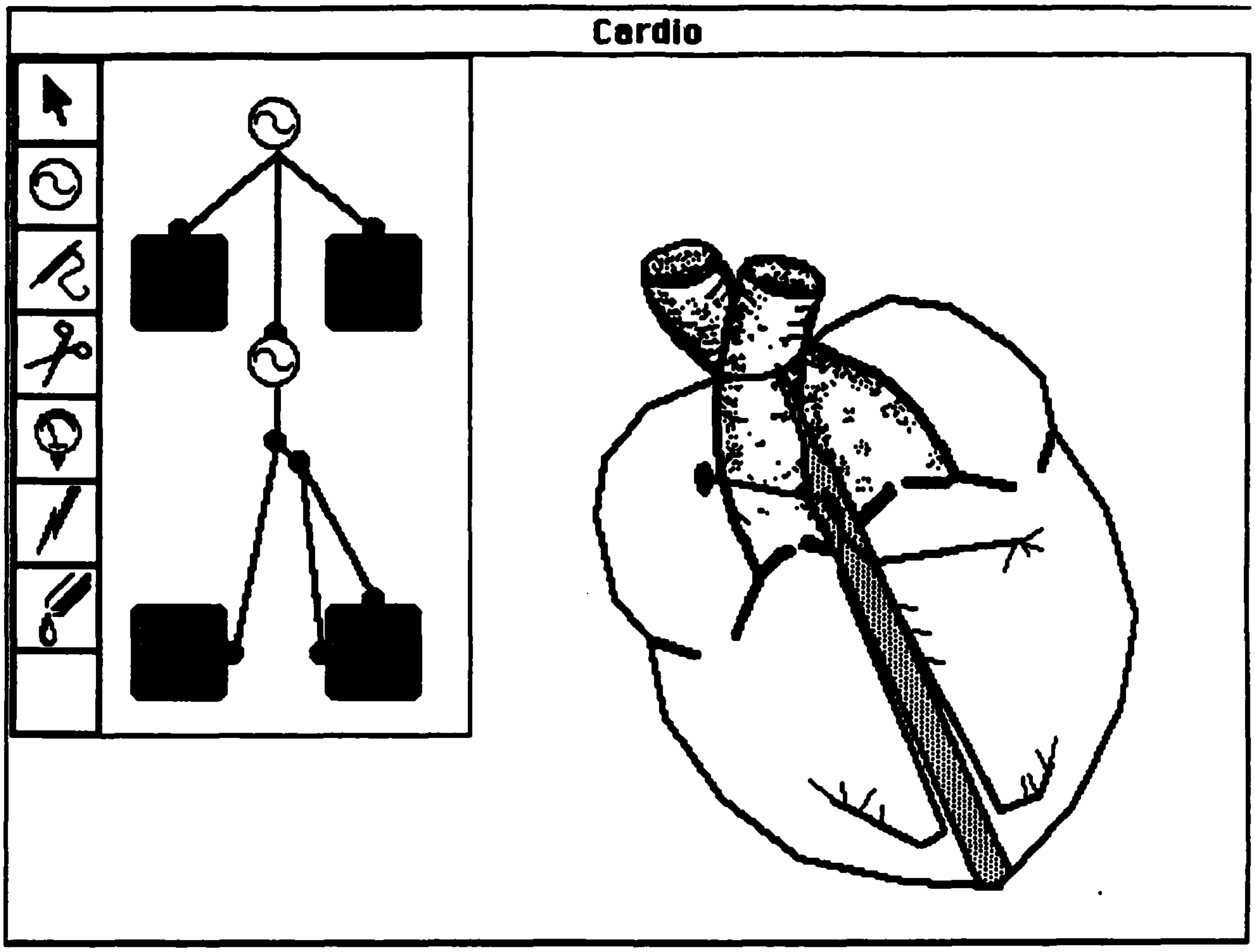


Figure 10

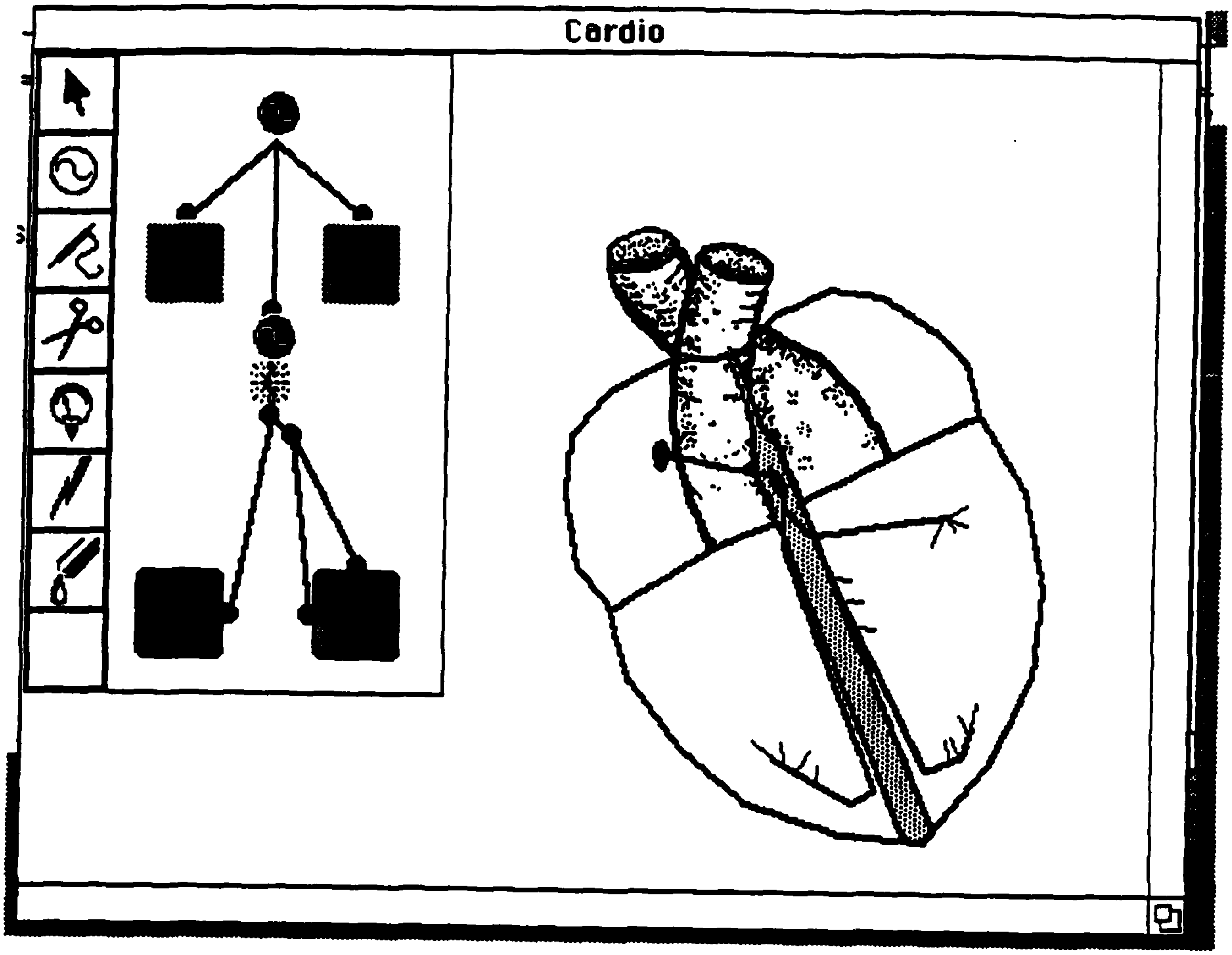
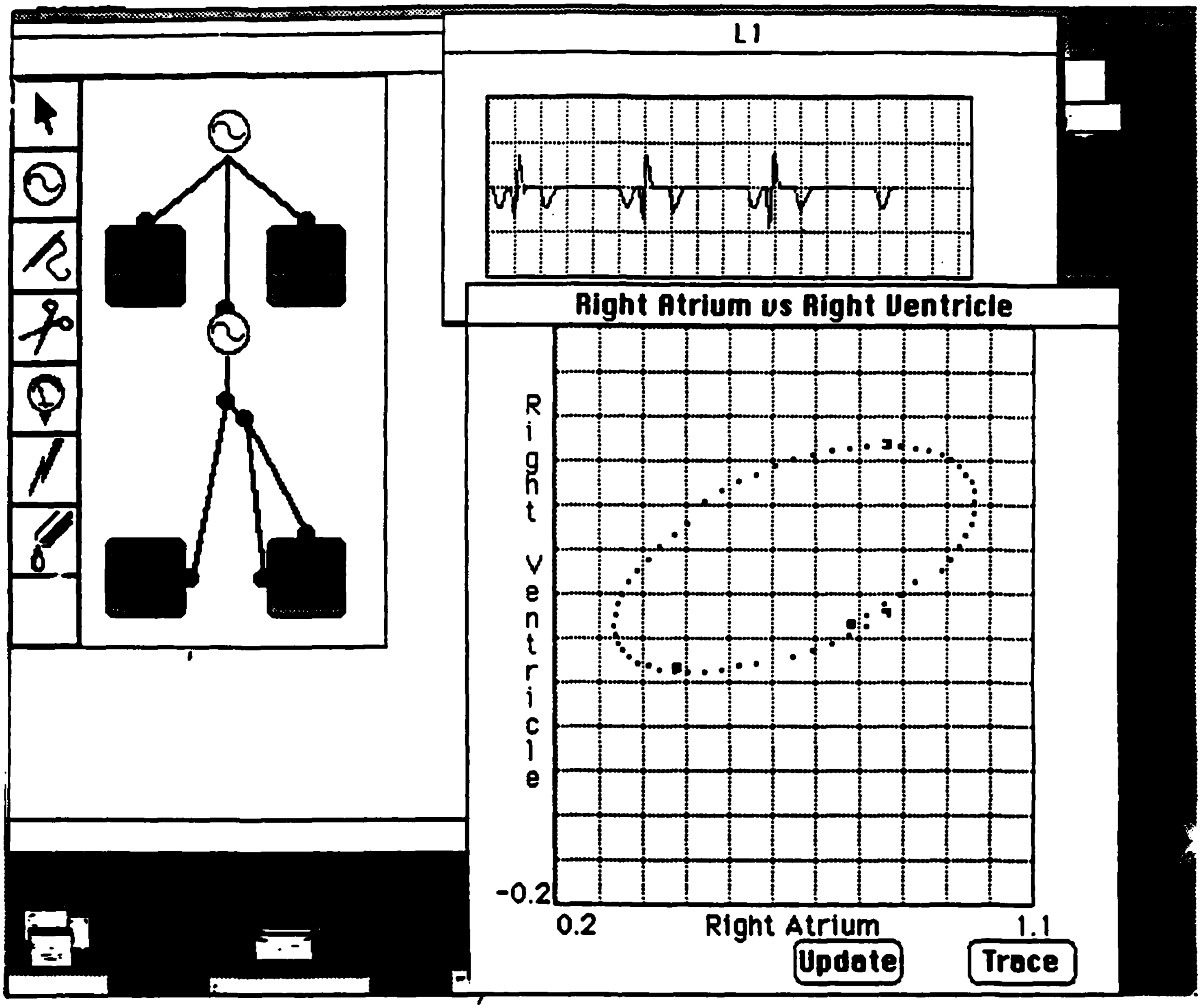


Figure 11



L1
Right Atrium vs Right Ventricle
Right Ventricle
-0.2
0.2
Right Atrium
1.1
Update
Trace

Figure 12

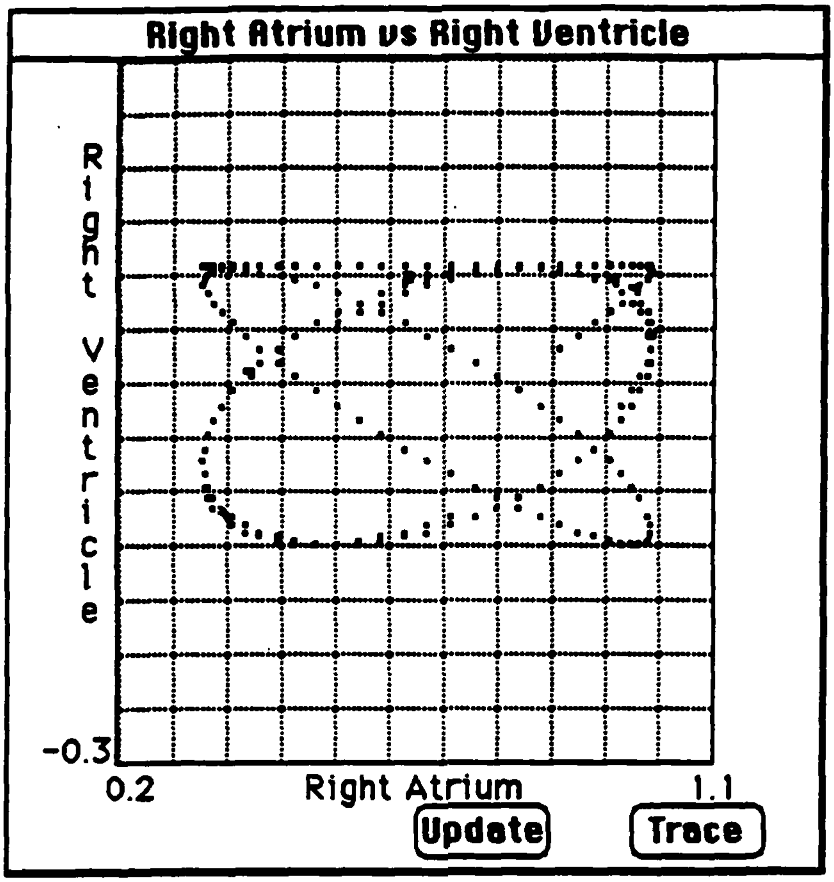

Right Atrium vs Right Ventricle
Right Ventricle
-0.3
0.2
Right Atrium
1.1
Update
Trace

Figure 13

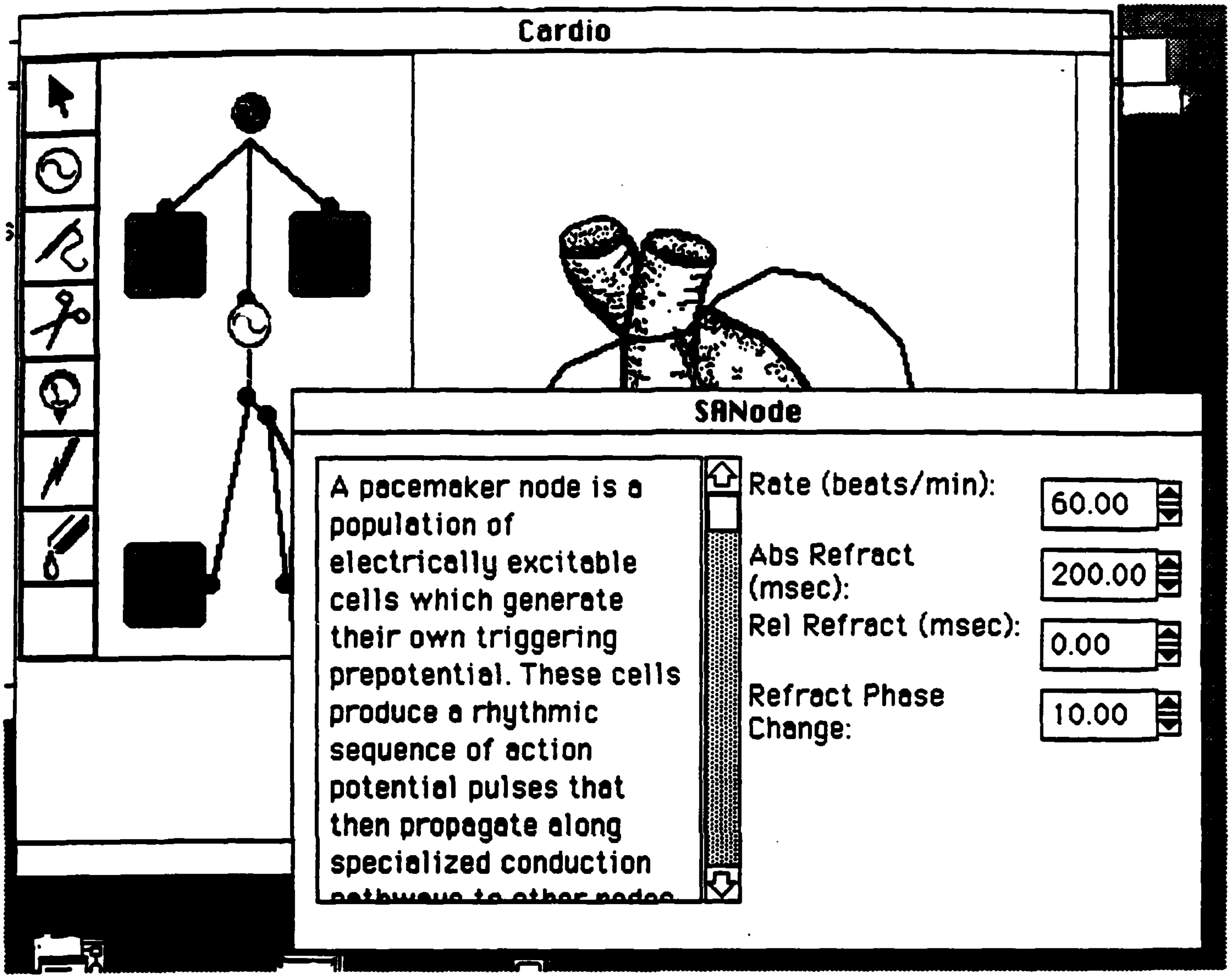

Cardio
SANode
A pacemaker node is a population of electrically excitable cells which generate their own triggering prepotential. These cells produce a rhythmic sequence of action potential pulses that then propagate along specialized conduction
Rate (beats/min):
60.00
Abs Refract (msec):
200.00
Rel Refract (msec):
0.00
Refract Phase Change:
10.00

Figure 14

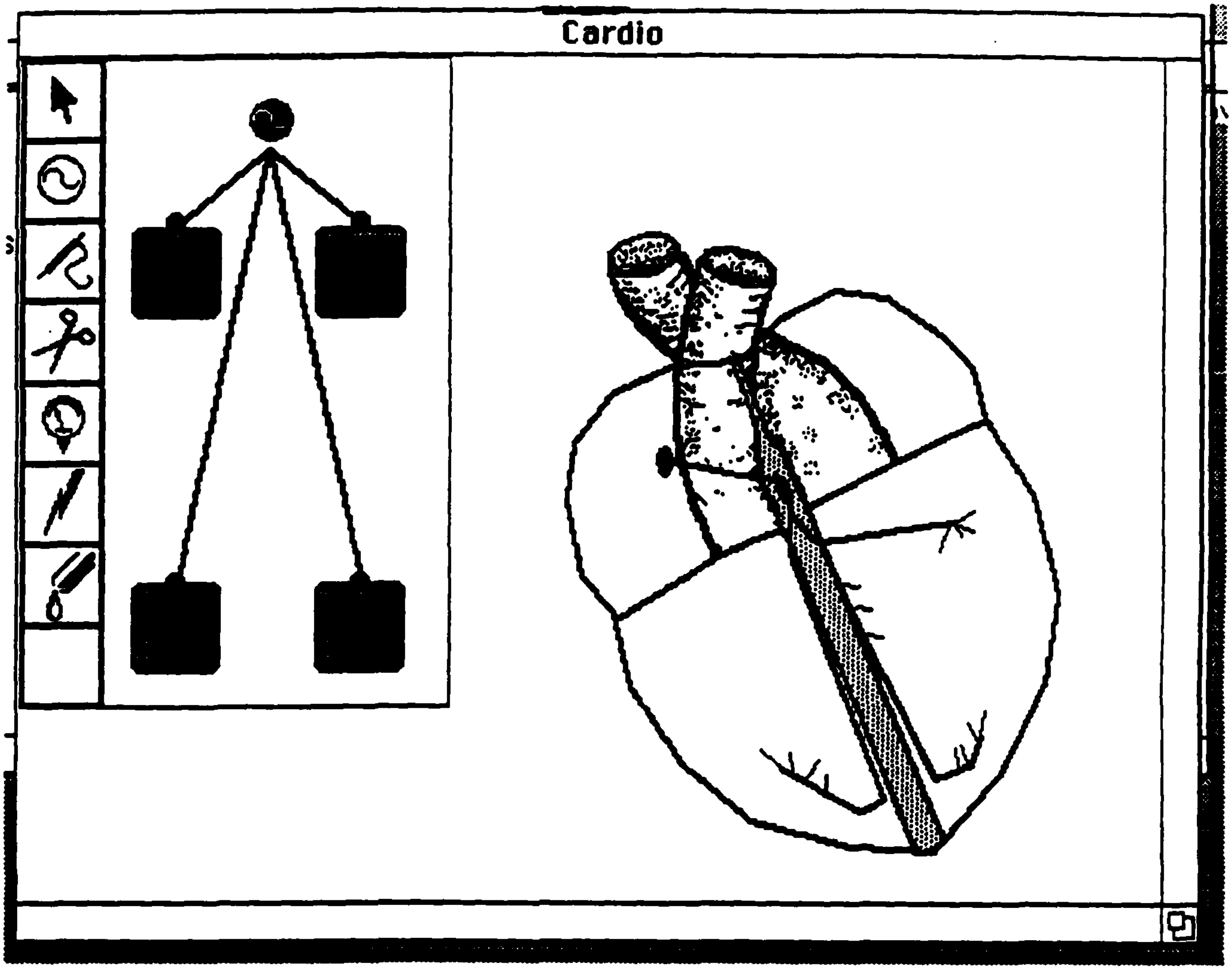


Figure 15

Assessing Programs That Invite Thinking

Susan R. Goldman

James W. Pellegrino

John Bransford

Vanderbilt University

Send correspondence to:

Dr. Susan R. Goldman
Box 45 GPC
Learning Technology Center
Vanderbilt University
Nashville, TN 37203

Assessing Programs that Invite Thinking

Our goal in this chapter is to discuss issues of evaluation that have arisen in the context of a problem solving series that has been developed by the Cognition and Technology Group at Vanderbilt's Learning Technology Center. The research and development of the series, called "The Adventures of Jasper Woodbury," began as an effort to offer an alternative to traditional classroom contexts where students often fail to see the relevance of what they are learning to real life (e.g., Cognition and Technology Group at Vanderbilt, 1990). A major goal of the series is to generate excitement about mathematics and science among middle school students (grades 5, 6, 7) and help them develop powerful skills of mathematical problem formulation and problem solving. A second goal is to help students see how content domains that are traditionally taught as separate "subjects" are actually integrated in the real world. Solving real problems often involves using math, science, geography, and economic concepts together. The Jasper problem solving series provides opportunities for students to experience such interdependence. A third goal of the series is to motivate students to become proficient in the "basic skills" of mathematics. We will say more about each of these goals in subsequent sections of the chapter.

The introduction of the series as part of the regular classroom curriculum has brought to the forefront critical assessment and evaluation issues. As evidenced by the other chapters in this volume, there is considerable interest in implementing new technologies in instructional settings and assessing the effects they have on instructional outcomes. What remains less clear is how to classify the various dimensions along which these technology implementations vary and their resultant impacts. This will be critical for sorting out ways in which new technologies can and should be implemented and realistic expectations about the outcomes that can or should be assessed. For example, a technology can be implemented to improve instructional management, or as a new way to deliver a current set of curricular materials, or as a means of providing new curricular content or meeting new curricular objectives. The extent to which a technology is designed to substitute for, enhance or change the extant curriculum will very much determine how we assess its impact. The Jasper series is designed to have the potential to relate to extant curriculum in all three of these ways. As a result, the assessment issues we discuss are directly related to the purposes for which the technology is used.

Just as a new technology can be implemented for multiple purposes, it is important to recognize that typical forms of assessment do not fall into a single functional category. Rather, they reflect multiple objectives, including (1) instructional management and monitoring, (2) program evaluation and accountability, and (3) selection and placement functions (Resnick & Resnick, in press). Because different objectives may demand different forms of instrumentation, the situation becomes complex very quickly if "double duty" is demanded of an assessment instrument originally designed to meet only a single objective. For example, there is the increasing realization that assessment for purposes of program evaluation and accountability has come to drive the nature of the curriculum and modes of instruction (e.g., Fredericksen & Collins, 1989; Resnick & Resnick, in press). In response to this realization the argument is made that curriculum reform can be brought about by changing the "accountability" assessments. The curriculum will change in response to changes in what the tests test. This realization suggests a fourth function for assessment: as an instrument of instructional reform. However, to accomplish curricular reform in this way, attention must also be directed at instructional management and monitoring issues.

One of our goals in this chapter is to illustrate the nature of the challenges we face in designing assessments that map onto new curricular goals such as emphasizing complex thinking and problem solving skills and integrating concepts across curricular areas. The challenges exist for both the instructional management and monitoring functions as well as the program evaluation and accountability functions. In short, even when the curricular materials and model are developed and implemented it is no simple chore to design a reasonable and workable set of assessment instruments. Yet it is critical that we provide some measures of student learning. We have found that the Jasper activities themselves suggest several useful evaluation paths that might be pursued. We will return to the assessment challenges following a discussion of the need for new approaches to instruction and the ways in which the Jasper series is designed to meet those needs.

The Need to Reassess Mathematics Curricula

It is useful to consider whether it us necessary to introduce an alternative mathematics curriculum, especially if that alternative creates the need to alter assessment and evaluation practices. There are multiple sources of evidence that there is a need to reassess and modify current mathematics curricula, particularly for problem solving. As a nation, we face the problem that the mathematical and scientific literacy of our students is falling short of what is needed for today's technological world. The National Science Board Commission on Precollege Education argues that the "basics" for the 21st Century must go beyond reading, writing, and arithmetic and must include communication, higher problem solving skills, and scientific and technological literacy. In addition, the Commission emphasizes that these new basics are needed by all students, ". . . not only the few for whom excellence is a social and economic tradition" (see also Shakhashiri, 1989). Results from national studies reinforce the need to increase achievement in mathematics and science (e.g., Fourth National Assessment of Educational Progress in Mathematics; Kouba, Brown, Carpenter, Lindquist, Silver, & Swafford, 1989). In addition, business leaders with whom we have discussed these issues agree that the "new basics" are needed for all our students, not simply a select few.

Along with the perceived need to increase levels of mathematics and science literacy, recent reports published by the National Council of Teachers of Mathematics (1989) and other professional organizations (AAAS, 1989; NRC, 1989) call for changes in the way mathematics is taught. These reports base their recommendations upon a number of factors, including (a) projected demographic changes in the nation's student population--the population from which tomorrow's workers will be drawn, (b) changing economic and technical needs of our society, (c) results from National Assessment of Educational Progress (NAEP) studies and cross-cultural comparisons that show serious deficiencies in the mathematical abilities of United States school children (Dossey et al., 1988; McKnight et al., 1987), and (d) a change in our views of how children learn mathematics.

There is now clear evidence that when learning takes place in meaningful contexts the result is knowledge that is longer-lasting and more accessible than knowledge learned in nonmeaningful settings (see for discussion Bransford, Sherwood, & Hasselbring, 1988; Cognition and Technology Group, 1990; Sherwood, Kinzer, Hasselbring, & Bransford, 1987). Furthermore, learning seems to take place best in familiar contexts and through active involvement on the part of the learner (references). The Jasper Series uses video-based narratives as the context for complex mathematical problem solving. It stands in sharp contrast to the materials used for topics in mathematics problem solving instruction.

Problems with Traditional Approaches

The traditional approach to teaching problem solving in mathematics involves the use of standard word problems such as those typically found in textbooks. There are several problems with this approach. One was captured by Gary Larson in his Far Side cartoon series. He created the concept of "Hell's Library" and populated it with nothing but book after book of mathematical story problems. Many students agree with his assessment and, as former students, many teachers do, too.

A second problem with traditional word problems is related to the fact that many students do not see them as real. The problems often seem contrived and arbitrary and have non-realistic goals. For example, one written problem we saw recently involved a trip to a haunted house. The setting seemed interesting. The problem posed to the students was: "There are 3 cobwebs on the first floor and 4 on the second floor. How many are there altogether?" This is hardly the kind of concern one would have when visiting a haunted house!

Data suggest that traditional word problems do not help students think about realistic situations. Instead of bringing real-world standards to their work, students seem to treat word problems as abstract situations and often fail to think about constraints imposed by real world experiences (e.g., Charles & Silver, 1988; Silver, 1986; Van Haneghan & Baker, 1989). For example, Silver (1986) noted that students who were asked to determine the number of busses needed to take a specific number of people on a field trip divided the total number of students by the number that each bus would hold and came up with answers like 2 1/3. The students failed to consider the fact that one cannot have a functioning 1/3 bus. Research also indicates that students have great difficulty with problems involving two or more steps and with problems that require reasoning skills (Kouba et al., 1989).

An additional problem with traditional word problems is that they **present** problems to be solved rather than help students learn to **generate and pose** their own problems. For example, imagine the task of going from one's house to a breakfast meeting at 8:30 in a new restaurant across town. What time should one leave? To answer this question, one has to generate sub-problems such as "How far away is the meeting?", "How fast will I be able to drive?", etc. The ability to generate the sub-problems to be solved is crucial for real-world mathematical thinking (e.g., Bransford & Stein, 1984; Brown & Walter, 1983; Charles & Silver, 1988; Porter, 1989; Sternberg, 1986). Word problems typical of text books do not develop such generative skills.

The problem of non-motivating and relatively ineffective materials for teaching mathematical problem solving has consequences that are particularly damaging during the middle school years. Educational experiences at this age may have important effects on students' interests and decisions about the kinds of courses to take in high school, which in turn affect career choices.

The Adventures of Jasper Woodbury, an Alternative to Traditional Mathematical Problem Solving Instruction

With the foregoing concerns in mind, we set out to create a context for mathematical problem solving that would be meaningful, realistic and motivating to students. The Jasper Woodbury series provides that context and is based on design principles that evolved over a 5 year period of research and development (e.g., Bransford et al., 1986, 1988, 1989;

Cognition and Technology Group at Vanderbilt, 1990; Sherwood et al., 1987, 1988; Van Haneghan et al., 1989; Young et al., 1989). Over the course of that time period, we have expanded the instructional functions for the Jasper series beyond complex mathematical problem solving. The Jasper series has evolved into a flexible instructional environment that can function on 2 additional levels:

• As an expandable curriculum in math, providing motivation and opportunities for anchoring component skills practice to a real problem.

• As a production tool for linking across the curriculum to other subject areas such as geography, science, history, reading, and writing.

The Jasper Series is video-based and can be used at several levels of technological sophistication. At the high technology end, the adventures are on videodiscs and the videodisc player is controlled via a microcomputer. This level of technology enables all 3 instructional functions. One step down involves videodisc-based adventures with the player controlled by a hand-held remote controller or by a bar code reader. This level enables two instructional functions (complex problem solving and extensions in math). At the least technologically sophisticated level, the adventures are on video tape and complex mathematics problem solving is the instructional focus.

Seven Design Principles for Adventures that Invite Thinking

The design principles for our problem solving series were selected because they enable teachers to create the kinds of mathematical problem solving experiences for students that are recommended in the NCTM curriculum standards (1989). This document contains a number of important suggestions for changes in the types of classroom activities and mathematical content to be emphasized in mathematics classes. Suggestions for changes in **classroom activities** include more emphasis on complex, open-ended problem solving; communication and reasoning; more connections of mathematics to other subjects and to the world outside the classroom; more uses of calculators and powerful computer-based tools such as spreadsheets and graphing programs for exploring relationships (as opposed to having students spend an inordinate amount of time calculating by hand). The design principles for our videos were developed to make it easier for teachers to provide opportunities such as these.

1. Video-Based Presentation Format. Although some excellent work on applied problem solving has been conducted with materials that are supplied orally or in writing (e.g., Lesh, 1981), we decided to use the video medium for several reasons. One is that it is easier to make the information more motivating because characters, settings, and actions can be much more interesting. A second reason for using the video medium is that the problems to be communicated can be much more complex and interconnected than they can be in the written medium--this is especially important for students who are below par in reading. Modern theories of reading comprehension focus on the construction of mental models of situations; students can more directly form a rich image or mental model of the problem situation when the information is displayed in the form of dynamic images rather than text (Bransford et al., in press; McNamara, Miller, & Bransford, in press). Teachers who have worked with our pilot videos have consistently remarked that our video-based adventures are especially good for students whose reading skills are below par.[1.] In addition, since there is a great deal of rich background information on the video, there is much more of an opportunity to notice scenes and events that can lead to the construction of additional, interesting problems - in other content areas as well as in mathematics (e.g., Bransford et al., in press).

2. Narrative Format. A second design principle is the use of a narrative format to present information. One purpose of using a well-formed story is to create a meaningful context for problem solving (for examples of other programs that use a narrative format, see Lipman, 1985; Voyage of the Mimi; 1984). Stories involve a text structure that is relatively well understood by middle school students (Stein & Trabasso, 1982). Using a familiar text structure as the context for presentation of mathematical concepts helps students generate an overall mental model of the situation and lets them understand authentic uses of mathematical concepts (e.g., Brown, Collins, & Duguid, 1989).

1.We will add a footnote noting that we can also use video in a way that enhances reading skills.

3. Generative Learning Format. The stories in the Jasper series are complete stories with one exception. As with most stories, there is setting information, a slate of characters, an initiating event and consequent events. The way in which these stories differ is that the resolution of the story must be provided by students. (There is a resolution on each disc, but students see it only after attempting to resolve the story themselves.) In the process of reaching a resolution, students generate and solve a complex mathematical problem. One reason for having students generate the ending--instead, for example, of guiding them through a modelled solution--is that it is motivating; students like to determine for themselves what the outcome will be. A second reason is that it allows students to actively participate in the learning process. Research findings suggest that there are very important benefits from having students generate information (Belli, Soraci, & Purdon, 1989; Slameka & Graf; 1978; Soraci, Bransford, Franks, & Chechile 1987).

4. Embedded Data Design. An especially important design feature of the Jasper series --one that is unique to our series and is instrumental in making it possible for students to engage in generative problem solving--is what we have called "embedded data" design. All the data needed to solve the problems are embedded somewhere in the video story. The mathematics problems are not explicitly formulated at the beginning of the video and the numerical information that is needed for the solutions is incidentally presented in the story. Students are then able to look back on the video and find all the data they need (this is very motivating). This design feature makes our problem solving series analogous to good mystery stories. At the end of a good mystery, one can see that all the clues were provided, but they had to be noticed as being relevant and put together in just the right way. The numerical information includes whole and mixed numbers and different forms of symbolic representation of quantity. The four basic arithmetic operations are required to solve the problem. Hence, in the context of a meaningful and complex problem, students understand the need for proficiency in adding, subtracting, multiplying and dividing. Among students for whom those skills are weak, we assume that discovering the usefulness of basic arithmetic skills is an important source of motivation to practice those very skills. (There is a real need for research on this issue.)

5. Problem Complexity. The Jasper videos pose very complex mathematical problems. For example, the first episode in the series contains a problem comprised of more than 15, interrelated steps. In the second episode, multiple solutions need to be considered by students in order to decide the optimum one. The complexity of the problems is intentional and is based on a very simple premise: **Students cannot be expected to learn to deal with complexity unless they have the opportunity to do so** (e.g., Schoenfeld, 1985). Students are not routinely provided with the opportunity to engage in the kind of sustained mathematical thinking necessary to solve the complex problem posed in each episode. The video makes the complexity manageable. We believe that a major reason for the lack of emphasis on complex problem solving (especially for lower achieving students) is the difficulties teachers face in communicating problem contexts that are motivating and complex yet ultimately solvable by students.

6. Pairs of Related Videos. The sixth design principle involves the use of pairs of related problem solving contexts. The cognitive science literature on learning and transfer indicates that concepts acquired in only one context tend to be welded to that context and hence are not likely to be spontaneously accessed and used in new settings (e.g., Bransford & Nitsch, 1978; Bransford, Franks, & Vye, 1989; Bransford, Sherwood, Vye & Rieser, 1986; Brown, Bransford, Ferrara, & Campione, 1983; Brown, Collins & Duguid, 1989; Gick & Holyoak, 1980; Salomon & Perkins, 1989; Simon, 1980). In the Jasper Series the content of each pair of episodes is similar. For example, the first two episodes deal with trip planning and require the use of distance, rate, and time concepts and their interrelationships, although the specific circumstances of the trips are quite different. A second pair of episodes deals with gathering, evaluating and assembling data into a defensible "business" plan for a project. Being able to apply principles learned in the first episode of each pair to the second allows students to experience the fact that problem solving gets easier the second time around. Students can also be helped to analyze exactly what they were able to carry over from one episode to another and what was specific to each and not generalizable.

7. Links Across the Curriculum. Each narrative episode contains the data necessary to solve the specific complex problem posed at the end of the video story. As well, the narration provides many opportunities to introduce topics from other subject matters. For example, in the trip planning episodes, maps are used to help figure out the solutions. These provide a natural link to geography, navigation, and other famous trips in which route planning was involved, e.g., Charles Lindbergh's solo flight across the Atlantic. We return to the topic of links across the curriculum in our subsequent discussion of the Jasper classroom.

Complex Mathematical Problem Solving in Trip Planning Adventures

Perhaps the best way to understand the design principles and the assessment issues they pose is to "walk" through the pair of episodes on trip planning. Each video has a main story that is approximately 17 minutes in length. The "end" of each video narration features one of the characters (Jasper in the first episode; Emily in the second) stating the problem that has to be solved; it is posed as a challenge and the students are to figure out the solution. (Note that a worked out solution is appended to the video story but students are not shown this until after they have solved the problem.) Please note that the verbal descriptions of these adventures fail to capture the excitement in them. In this case, a video is truly worth a thousand words.

Journey to Cedar Creek. This episode opens with Jasper Woodbury practicing his golf swing. The newspaper is delivered and Jasper turns to the classified ads for boats. Jasper sees an ad for a 56 Chris Craft cruiser and decides to take a trip to Cedar Creek where it is docked. He rides his bicycle to the dock where his small "row" boat, complete with outboard motor, is docked. We see Jasper as he prepares for the trip from his dock to Cedar Creek: He is shown consulting a map of the river route from his home dock to the dock at Cedar Creek, listening to reports of weather conditions on his marine radio, and checking the gas for his outboard. As the story continues, Jasper stops for gas at Larry's dock. Larry is a comical looking character who knows lots of interesting information. For example as he hands Jasper the hose on the gas pump, he just happens to mention all the major locations where oil is found. When Jasper pays for the gas, we discover the only cash he has is a $20 bill. As

Jasper makes his way up river, he passes a paddle-wheeler, a barge and a tug boat and some information is provided about each of these. Next, Jasper runs into a bit of trouble when he hits something in the river and breaks his sheer pin. He has to row to a repair shop where he pays to have the pin fixed. Later, Jasper reaches the dock where the cruiser is located and meets Sal, the cruiser's owner. She tells him about the cruiser and they take the boat out for a spin. Along the way, Jasper learns about its cruising speed, fuel consumption, fuel capacity and that the cruiser's temporary fuel tank only holds 12-gallons. He also learns that the boat's running lights don't work so the boat can't be out on the river after sunset. Jasper eventually decides to buy the old cruiser, and pays with a check. He then thinks about whether he can get to his home dock by sunset. The episode ends by turning the problem over to the students to solve.

It is at this point that students move from the passive television-like viewing to the active generation mode discussed earlier. They must solve Jasper's problem. Students have to generate the kinds of problems that Jasper has to consider in order to make the decision about whether he can get the boat home before dark without running out of fuel. The problem looks deceivingly simple; in reality it involves many subproblems. But all the data needed to solve the problem were presented in the video. For example, to determine whether Jasper can reach home before sunset, students must calculate the total time the trip will take. To determine total time, they need to know the distance between the cruiser's and Jasper's home dock and the boat's cruising speed. The distance information can be obtained by referring to the mile markers on the map Jasper consulted when he first began his trip. The time needed for the trip must be compared to the time available for the trip by considering current time and the time of sunset, information given over the marine radio. The problems associated with Jasper's decision about whether he has enough fuel to make it home are even more complex. As it turns out, he does not have enough gas and he must plan for where to purchase some--at this point money becomes a relevant issue. In this manner, students identify and work out the various interconnected subproblems that must be faced to solve Jasper's problem.

<u>Rescue at Boone's Meadow.</u> The second Jasper episode on trip planning is equally complex and involves planning for a rescue. The story

revolves around Emily, a friend of Jasper and Larry. Emily is learning to fly an ultralight; her instructor is Larry. When the story begins, we see Larry describing the plane. He tells Emily about the weight limitations of the plane, its fuel consumption, fuel capacity, airspeed and so forth. Some days later, Emily makes her first solo flight, after which she, Jasper, and Larry celebrate over dinner. While eating, Jasper tells them of his plans to hike into the wilderness to go fishing. The next scene shows Jasper at this remote fishing area. The tranquillity of the scene is disturbed by the sound of a gunshot. Upon investigating, Jasper finds that an eagle has been shot and wounded. Jasper immediately radios Emily for help.

Again, at this point in the story students move from passive television-like viewing to an active generation mode. They must decide the fastest way for Emily to get the eagle to a veterinarian. There are multiple vehicles, agents, and routes that can be used, subject to the constraints introduced by the terrain and capacities of the various vehicles and available agents. Like Jasper's river trip problem, the solution involves a multi-step, distance-rate-time problem. It thus allows students to use the general schema of the Journey to Cedar Creek episode. In addition, the rescue problem involves generating multiple rescue plans and determining which is the quickest.

The episodes in the Jasper Series involve students in planning complex problem solutions (including problem formulation and problem posing); information search, retrieval and organization; and monitoring and evaluation processes. These cognitive activities are those mentioned in conjunction with critical thinking skills (e.g., Bransford, Goldman, & Vye, in press; Resnick & Klopfer, 1989).

Additional Instructional Functions for the Jasper Adventures

In addition to its function as a context for engaging in complex mathematical problem solving, the Jasper episodes can be extended to other areas in the mathematics curriculum and to other curricular areas such as English, Science, and Social Studies. For example, in the context of the trip planning videos, we have developed materials that anchor practice on measurement concepts to the Jasper episodes. To develop fluency with units of measurement, students are shown various objects and people who were seen in the episode and they are asked to estimate the size. The task involves discriminating among common units of measurement, e.g., pounds, ounces, tons; inches, feet, weight. To develop proficiency at applying the Jasper problem solution, the parameters can be changed. For example, by changing the fuel capacity of the temporary fuel tank, Jasper may be able to make it home without refueling. To develop proficiency with trip planning, there are available a number of "perturbed" mini-adventures that utilize the trip-planning schema as well.

The Jasper episodes were also designed to provide natural contexts for exploring science-related concepts such as (a) the density of metals (when determining materials for building boats and planes); (b) density of liquids (when considering the amount and effects of payloads); (c) exploring advances in communication and weather prediction (for boating, flying and exploration). We focused on the general issue of planning for trips (e.g., going down the river or flying to rescue the wounded eagle) because general "trip planning schemata" are applicable to a wide range of topics such as Lindbergh's flight across the Atlantic, Admiral Byrd's exploration of the Antarctic, the NASA space flight to the moon, etc. To capitalize on these linkages across content areas, we have a prototype version of "Publisher" software that makes use of hypermedia.

Hypermedia can be defined as the linkage of text, sound, video, graphics, and the computer in such a way that access to each of these media is virtually instantaneous. Through hypermedia, fundamental thinking and learning activities such as elaboration and flexible coding are facilitated because multiple modalities and symbolic systems can be represented and multiple connections among information can be established. The "Publisher" software provides easy access to hypermedia databases and calculating tools that allow students to explore interesting facts, ideas, and concepts in these domains and to understand how mathematics relates to these other areas.

The databases constructed with the "Publisher" provide students with an opportunity to explore scientific, geographical, and historical information related to the Jasper adventures and to use this information to solve real-world problems that people such as explorers and others throughout history have had to solve. As a simple illustration, the database for Episode I includes historical information relevant to life during the times of Mark Twain. When studying Mark Twain's world, it is very instructive for students to see how plans to go certain distances by water in Journey to Cedar Creek would be different if the mode of travel were steamboat or raft. A three hour trip for Jasper by motorboat would have taken the better part of a day on Huckleberry's raft. This means that drinking water, food, and other necessities would need to be included in one's plans.

It is noteworthy that students like to find their own problems and issues and add this information to the database. Their contributions can contain information about who submitted them, so students can be published in the school (or state or national) database. Our decision to begin the Jasper series with an emphasis on general principles of trip-planning makes it easier to extend the mathematical thinking to a variety of domains (e.g., space explorations, historical expeditions such as Admiral Byrd's trip to Antarctica, Lindbergh's flight to Europe, etc.). We encourage students to explore specific segments on other videodiscs, such as those produced by National Geographic, that chronicle historic expeditions and other adventures that show evidence that the adventurers had to plan in similar ways to the characters in the Jasper adventures.

Assessment Challenges

From observations, anecdotes, and personal reports we know that reaction to the Jasper Series is extremely positive. When children work with Jasper their involvement and enthusiasm are evident. Indeed, attitude data we have collected indicate that students of all ability levels enjoy solving the Jasper problems and would like to solve additional ones (VanHaneghan, Vye, et al., in preparation). Teacher reports indicate that children are excited by the adventures and interested in solving them. Teachers themselves have been extremely enthusiastic about the cross curricular links. They see many ways to extend the Jasper episodes into many areas of the curriculum. For example, at a recent Training Institute, two dozen teachers worked with the Jasper Publisher and created cross-curricular links. Each group of two came up with a different extension.

It is evident from the varied nature of the data, as well as from the variety of ways in which Jasper can be used in the classroom, that there are multiple assessment challenges to be addressed. The most appropriate assessment techniques can be expected to vary with the *instructional function being evaluated.* Add to this the issue of the purpose for which the assessment is undertaken and the complexity of the assessment challenges becomes clear. Clearly, the basic question for Jasper assessment is What are the effects of using Jasper? In the remainder of the chapter, we discuss our current thinking about approaches to these assessment challenges. We have organized the discussion around the three instructional functions that the Jasper adventures may play. For each function, we discuss assessment for purposes of characterizing student learning in research contexts and in classroom contexts.

Assessment of Complex Mathematical Problem Solving.

We noted earlier that a major goal of the Jasper series is to help students learn to pose, formulate and solve complex problems that are similar to those often encountered in everyday settings. Each Jasper adventure ends with such a problem and we have also developed sets of analogs that provide extra practice for each adventure. Clearly, in order to assess the effects of this aspect of the curriculum on students' thinking, we need to measure the degree to which they transfer to new types of problem solving tasks. For purposes of our research, the field of cognitive science provides useful information about ways to assess complex problem solving--at least for individuals. Verbal protocals are often regarded as benchmark measures of these processes. Empirical investigations conducted under controlled conditions indicate that solving the first Jasper adventure does indeed improve children's complex problem solving skills (Cognition and Technology Group at Vanderbilt, 1990; Van Haneghan, et al., in press; Young, et al., 1990). In those studies we have used verbal protocols from individual interviews with children. We regard these as benchmarks against which to evaluate other forms of assessment. The Challenges are to (1) develop surrogates of the interviews that are less time consuming to administer yet maintain the validity and reliability of the interviews; (2) determine cognitive process measures that are appropriate to group problem solving.

A major reason for attempting to develop surrogates of our verbal protocol interview tests is that the latter are not feasible to administer under normal classroom conditions. This presents a potential problem because measures of student mastery of the processes involved in complex mathematical problem solving are critical to instructional management and monitoring. If teachers do not have some measure of how well students are doing in complex problem solving, they will not be able to evaluate their own teaching strategies and whether they need to change them, nor the degree to which additional instructional time is needed and for whom.

One approach that we have taken in this regard is to attempt to develop paper and pencil measures that provide data that are consistent with our interview measures. Because one of the major goals of our interviews is to measure problem formulation or generation, we cannot simply administer multiple choice tests such as "Which of the following questions does Jasper need to ask himself in order to make his decision: When is sunset? How fast is the boat? How wide is the river? etc." The ability to **recognize** relevant questions is not the same as the ability to **generate** them on one's own. Therefore, there are many constraints on the nature of the tests that we can create.

At present we are experimenting with a paper and pencil test that involves a generative component (students must write down relevant questions to be solved) and an explanatory component (students must explain why someone who was attempting to solve Jasper's problem had carried out certain calculations). Following instruction on a Jasper adventure (when we try to assess without instruction we run into floor effects), the same students are being given a verbal protocol interview test and the paper and pencil test, with order of testing counterbalanced. Data such as these will allow us to measure the relative comparability of each type of test.

Several members of our group, especially Mike Young (now at the University of Connecticut) and Jim Van Haneghan (now at University of Northern Illinois) have been experimenting with ways to use technology to assess complex problem solving. Our Jasper adventures can be controlled through a "map" controller that allows easy access to any scene on the disc. The map, shown in Figure 1, is exactly like the map that Jasper used in the video. it includes all of the locations and events that were depicted in the Journey to Cedar Creek Adventure. When a particular location is "clicked," the various video scenes that occurred at that location can be accessed. For purposes of assessing students mastery of complex problem solving, students can be asked to go back and find the information necessary for solving Jasper's problems. By keeping track of where on the map students look and the order in which they access different locations and scenes, we hope to get a good measure of problem generation that is not contaminated by the use of multiple choice questions that themselves can act as cues.

Insert Figure 1 about here

Assessing the complex mathematical problem solving skills of students is most useful for purposes of formative assessment and instructional management. Verbal protocols and surrogate tests of problem solving such as the ones that we have described in this section can be used to assess **mastery** of the problem solving processes relevant to a specific Jasper adventure as well as **transfer** to new problems. A major goal of these assessments is to identify children who need more work before progressing further in the Jasper series. For transfer assessment we have devised tests that involve new complex problems that systematically differ from specific Jasper adventures. These Jasper problem "analogues" are also one of the ways in which the Jasper adventures extend into other areas of the math curriculum and are discussed more fully in that section.

Measures of Group Problem Solving

We are also attempting to assess the effects of using Jasper adventures in the context of group problem solving. Here we encounter a number of theoretical and methodological issues that have important implications for the assessment strategies to be used.

Consider the question of whether group problem solving using Jasper *is superior to individual problem solving. If we show students a Jasper* adventure and then ask them to solve it either individually or in groups, *the groups almost always do better. Many other studies show similar* effects (e.g., Johnson, Maruyama, Johnson, Nelson, & Skon, 1981; Slavin, 1984). But what do these results *mean?* Work being conducted by a member of our Center, Brigid Barron, is designed to test several models of why groups may perform better than the average of a group of individuals. One class of models describes group performance by one member's level of performance. There are two possibilities here: the most competent member model and the least competent member model. According to these models, group performance is either as good (or as bad) as the strongest (or the weakest) member of the group. A second class of models holds that the performance of the group exceeds what any individual member of the group could achieve. The pooling of abilities model is one example of this type of model: individual group members solve different portions of the problem and "pool" their efforts for the complete solution. An additional model in this class is the synergistic model. This model assumes that exchanges among group members promote thinking in ways that would not have occurred had the individuals worked alone. This exchange process helps all components of the problem solving process (Barron, in progress).

In order to assess the effects of group problem solving on individual students, it is necessary to test the abilities of individuals to solve new problems. A relatively small number of studies have addressed this question (e.g., Larson, Dansereau, O'Donnell, Hythecker, Lambiotte, & Rocklin, 1985) but several members of our Center are currently exploring it. They are also looking at the effects of "scripting" group interactions on the nature of the group problem solving process and on the effects on both group and individual problem solving.

It seems clear that each of the models of group problem solving that were discussed above (e.g. the most or least competent member model, synergistic model, etc.) may be operative under some circumstances. For example, empirical studies that have compared heterogenous with homogenous ability groups (e.g., Goldman, 1965; Webb, Ender, & Lewis, 1986) indicate that outcomes are mediated by the nature of the interactions that occur in the group. This being so, instructional management and monitoring purposes are best served by assessment techniques that enable teachers to characterize the group interaction, coupled with instructional techniques, such as scripting, for directing those interactions.

Assessment of Extensions to Other Areas of the Math Curriculum

The Jasper adventures can serve as anchors for work on aspects of the math curriculum not directly involved in solving the specific problem posed by a particular adventure. One important aspect of the math curriculum is the need to develop fluency with basic concepts and procedures. There is, however, a strong need for greater theoretical consistency regarding the concept of fluency, especially in different skill areas, and in the linkages between fluency on component skills and complex thinking and problem solving.

From a research perspective, we know how to measure fluency of basic concepts (e.g., 5+7 = 12) and we know that increased fluency on component skills can be achieved via practice (Goldman, Pellegrino, & Mertz, 1988; Goldman, Mertz, Pellegrino, 1989; Hasselbring, Goin, & Bransford, 1989). We also know that if students practice procedural algorithms, such as those used in long division, their accuracy and efficiency improve (Sherwood, et al. 1990). We have yet to examine the development of fluency with complex problem solving procedures. The Jasper adventures can function as an anchor for fluency training in all three areas of mathematics. For example, they can serve as an anchor for practice with the basic concepts introduced in the video, such as units of measurement and conversions among them (e.g., inches, feet, yards) as well as equivalences among different symbolic representations for quantities (e.g., decimals, fractions, mixed numbers, percentages). Procedural algorithms for specific calculations are used in the Jasper problem solutions; students may be motivated to become more proficient with these procedures when they see them used to solve a meaningful problem.

We can also use the Jasper adventures to anchor practice with complex problem solving procedures by creating variants of the problem presented in the video. For example, a second complex problem can be easily created by altering the original problem constraints, e.g., the cruiser travels at a slower (or faster) rate or the time of sunset changes, etc. Students need to use the same sets of planning and solution procedures but the specific calculations are different; sometimes changing the constraints also makes some steps unnecessary. We are working on computer as well as paper and pencil analogues and are developing assessments of these that focus on problem generation and solution explanation skills. Performance on these Jasper isomorphs can serve practice and instructional purposes as well as serving as assessment techniques in their own right. Such performance based assessment provides an index of student learning useful for instructional management and monitoring purposes and may hold promise for program evaluation and accountability purposes.

Clearly, an emphasis on the importance of fluency is not new. In the area of reading, for example, Laberge and Samuels (1974) argued that accuracy in reading words should not be confused with fluent access to their pronounciation. The same is true in the mathematics domain (e.g., Goldman & Pellegrino, 1987; Kaye, 1986). Despite the importance of fluency in the theoretical literature, many curricula do not emphasize it. Instead, students are often introduced to new ideas for a short period of time; instruction then shifts to other ideas and there is little chance to become fluent at accessing basic concepts and skills. Although practice exercises are a common classroom activity, the emphasis is usually on accuracy and not on fluency. If instructional planning decisions are made only on the basis of accuracy, there may seem to be no need for further practice.

The importance of focusing on fluency in the area of mathematics is illustrated by data from a cross-sectional study of math delayed and normally achieving students (Hasselbring, Goin & Bransford, 1988; Hasselbring, Goin, Alcantara, & Bransford, 1990). The study assessed students' abilities to add specific facts such as 5 + 8 = ? Figure 2a shows that when accuracy is measured, math delayed and normally achieving students perform at similar levels by about age 9 (roughly fourth grade). In contrast, Figure 2b shows the results for fluency: the gap between the two groups increases with age.

Insert Figure 2 about here

In related work, we have also found that the traditional practice activities associated with many curricula involve work on specific skills and procedures in isolation or outside of contexts where the information might be needed. For example, the Mastering Fractions videodisc program (need ref.) involves a series of lessons that helped students become very proficient at working with fractions problems such as 1/2 + 1/4 = ? , 1/3 X 1/4 = ? (e.g., Sherwood, et al., 1990). Nevertheless, when students were asked to use their knowlege of fractions to solve word problems, they did poorly and no better than a control group (Sherwood, et al.,1990). Basically, the students' knowledge of fractions remained inert. We assume that the opportunity to move back-and-forth between Jasper problem solving environments and fluency exercises on basic skills will promote the kinds of knowledge representations that appear to underlie expert performance and that avoid the inert knowledge problem (e.g. see Bransford & Vye, 1989; Chi, et al.,1989). In addition, if students achieve levels of proficiency that free attentional resources from the "basics", they can reallocate these to more complex thinking required for solving real world problems.

The assessment challenges with respect to fluency are (a) to measure both the speed of accessing concepts and skills as well as the degree to which students understand such concepts and are able to relate them to real-world problem solving situations; and (b) to deliver this information in forms that are functional for instructional management and planning purposes and perhaps for program evaluation and accountability.

Assessment of Cross-Curricular Extensions

In addition to the focus on complex problem solving and fluency, the Jasper adventures provide a context for helping students appreciate the the relevance and interrelatedness of various curriculum areas. Rather than provide these links in a text-book like fashion, we decided to create conditions that would motivate students to provide them. In an earlier section of this chapter, we referred to our HyperCard "Publisher" environment and noted that, unlike traditional report formats that present information in a linear order, hypermedia allows for "many branching" links among concepts. The "Jasper Publisher" is a tool to be used by the students. Using this software, students produce relational databases instead of 10 or 20 page papers. This activity should help them learn to notice relevant issues to be explored, learn how to find information to explore the issues, and learn to communicate the results of their findings. The Jasper Publisher program also allows students to create multi-media presentations to show others about the results of their research.

There are many questions with respect to the assessment and evaluation of student-generated multi-media projects. We have considered several possibilities for the skills and processes we might evaluate. We might look at

- the skill of posing interesting questions
- how individuals frame issues
- effectiveness and efficiency of information search and retrieval skills

The technology itself may facilitate certain forms of assessement of the aforementioned skills. *For example,* graph theory might be used to analyze the relational databases that students produce in terms of the number and nature of the links among different entries. We might also track the production process by analysis of "dribble" files or traces of the students production process. (We note the impracticality of this on any large scale.) Alternatively, we might assume that the act of generating the database is a sufficient performance index. These products could comprise student portfolios. The quality of the projects should improve over time and should show evidence of students' increasing familiarity with multi-media publishing. Familiarity with this new media should affect how students think about new reports that they choose or are asked to write.

Guidelines for other forms of assessment of the aforementioned skills and processes come from the "Young Sherlock" program that has been studied by members of our Center (e.g. Bransford et al.,1989; Cognition and Technology Group at Vanderbilt, 1990, Kinzer et al.,1990; Risko et al.,1990; Rowe, et al.,1990). The Sherlock project used the videodisc "The Young Sherlock Holmes" as an anchor for literacy instruction but its use also involved many cross-curricular links to history, geography, and science. Thus, it has a number of overlaps with respect to the cross-curricular extensions of Jasper.

Classroom ethnography revealed that one important effect of the Sherlock program was an extremely high number of student-generated questions (e.g., see Rowe et al.,1989). A major reason for this high rate of student-initiated questions was that all students shared an interesting macro-context yet all were allowed to choose their own specific points to query and explore further. Students quickly began to notice many features of the videos that they wanted more information about, and this frequently led to lively discussions and searches for relevant information. We expect similar types of activities with Jasper and could observe this through ethnographic studies such as those used in the Sherlock project. Needless to say, however, this is expensive and time-consuming research to perform.

An alternative to classroom ethnography is to show groups of students new sets of videos and ask them to state what they noticed in these videos and why it might be important to explore these issues in more detail. In research in the Sherlock project we found evidence that experimental students did better at this task than those in a control group (e.g., Vye et al.,1990). A related assessment task is to provide students with a topic to explore and ask them to show how they would search for relevant information. The more that students have had to search (e.g. through the library, through national databases) in order to retrieve relevant information, the better thay should be at information search.

A potential danger of database production is that the database produced by one student can very easily become the decontextualized and inert knowledge of another. To forestall this outcome, we encourage students (and teachers) who create multi-media presentations to pose interesting questions and challenge other students to accept the challenges. The ability of students to pose interesting questions given an assigned topic provides valuable information about what they have learned. Challenge questions also become a way of closely linking instruction and assessment an issue we turn to by way of concluding this chapter.

Assessment-Based Instruction

Challenge questions are an approach to instruction that makes use of information about assessment to drive teaching and learning. We call this approach our Jasper "Challenge Series". To date, it is simply an idea rather than something that we have had a chance to test. The Challenge Series is based on the idea of presenting interesting challenges that **students have the opportunity to prepare for ahead of time.** After a set amount of time to prepare for the challenge (several weeks, perhaps,) the students watch a telecommunications-based broadcast (something like a game show) that is transmitted to a number of schools across the state. All the students in a class team up to attempt to come up with answers to questions asked by the broadcasters (the answers can be communicated quickly through telephone lines); classes of students can then compare their answers with one another. Each class of students also gets the chance to ask several questions of its own--questions that get rated by all the other participants in terms of the degree to which they are "fair" and "astute". This allows students to learn to ask relevant questions about areas. The idea of participating in a live game-show format is very motivating to students and encourages them to prepare even better for the next round.

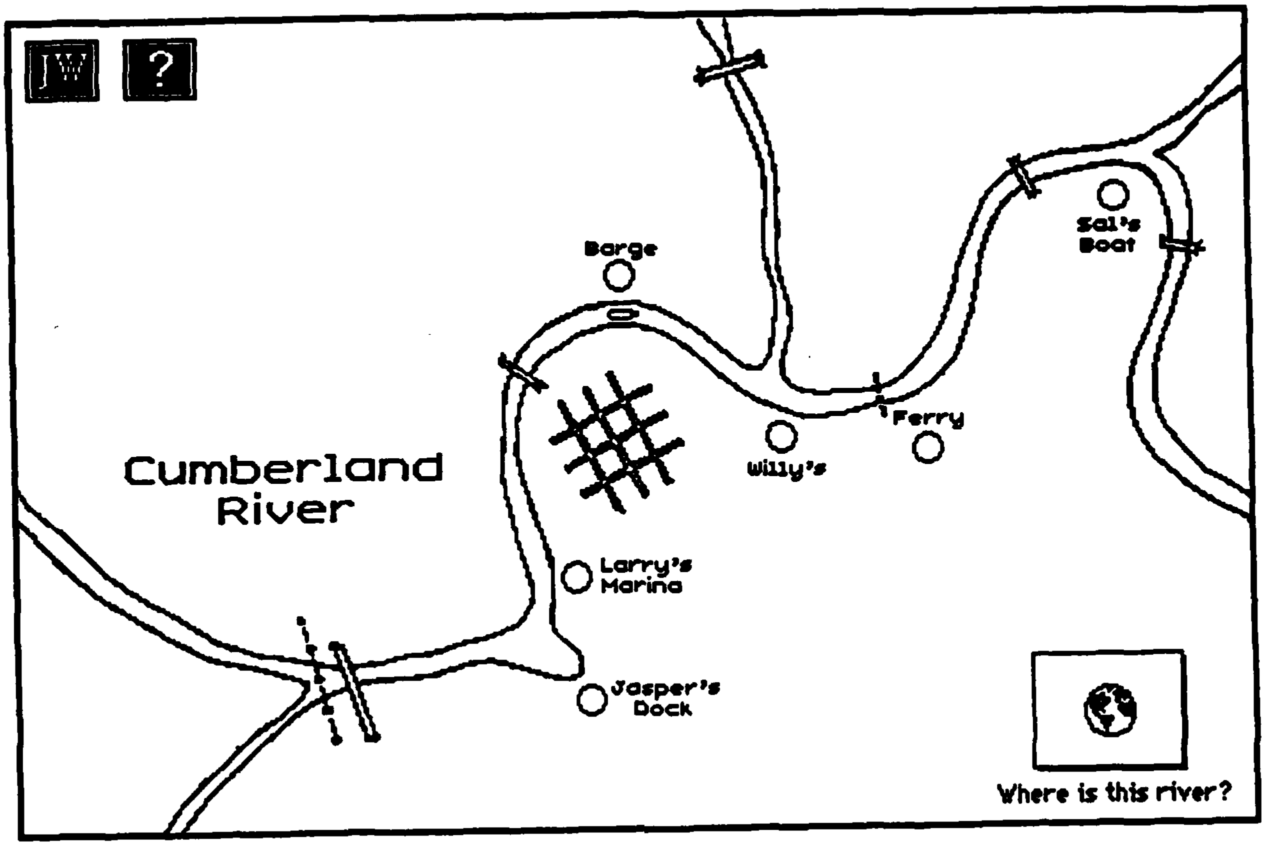
In addition to on-line scores for each class of students, classes will later receive off-line scores that reflect the average score of each individual in the class on the challenge series (each student will answer individually before then discussing answers as a group). Students can then compare their average class scores with those of other schools. The desire to get a high average score for one's class provides motivation for students to try to help one another learn during the preparation period. This provides a spirit of collaboration that allows classes to learn to function as teams.

To have broad effects, challenges need to be given a number of times during the year--not simply once. As students and teachers learn about the nature of the first challenge, they will be in a better position to figure out how to prepare for the next ones. Thus, we expect challenge scores to continually improve. Students should become more efficient at dividing tasks into sub-tasks and working as cooperative groups.

The basis of the Challenge Series is essentially that students learn to study differently depending on the test and that teachers tend to teach to the tests. Many educators argue that the problem of "teaching to the test" is not a problem if the tests demand important sets of skills and knowledge. (e.g., Fredrickson & Collins, 1989; Resnick & Resnick, in press). If the tests demand rigorous problem solving, for example, they seem to be worth teaching to. It is when tests demand only the rote memory of isolated facts or procedures that it seems harmful for teachers to teach to them. Many educators also argue that the creation of new and more challenging tests may be the most efficient and effective ways to spark a change in the kinds of teaching that typically takes place in classrooms. Our idea for a challenge series is one answer to the challenge of how to create tests that are worth teaching to and that can be administered on a large scale. The Challenge Series is also desiged to overcome one additional problem with testing; namely that being tested is usually a noxious event. In contrast, the challenge series TV programs could be a great deal of fun (we will use people who can make them this way).

Summary and Conclusions (To Be Added)

FIGURE 1



Barge
Sal's Boat
Cumberland River
Ferry
Willy's
Larry's Marina
Jasper's Dock
Where is this river?

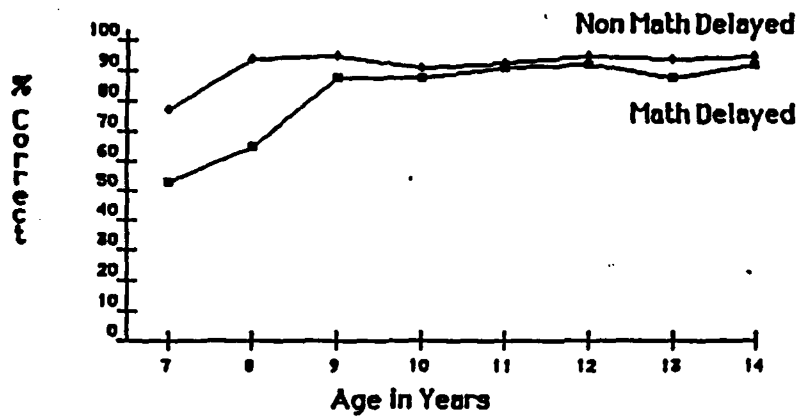


Figure 25. Comparison of response accuracy by age for math delayed and non-math delayed students.

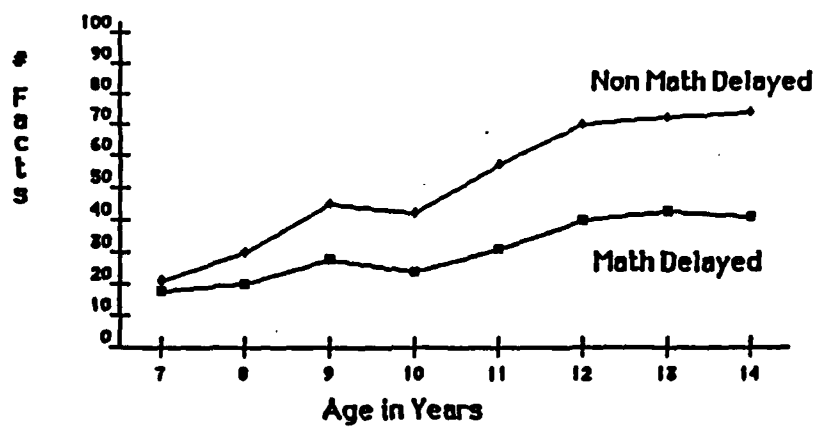


Figure 26. Comparison of the number of fluent facts by age for math delayed and non-math delayed students.

Interactive Technologies and the Assessment of Learning

Jan Hawkins, Allan Collins & John Frederiksen

Paper prepared for the UCLA Conference on
Technology Assessment: Estimating the Future
September, 1990

Our ways of assessing student learning in schools throughout the country are now beginning to undergo the scrutiny required to undertake major redesign. A newly designed assessment system must accurately measure and promote the complex thinking and learning goals that are known to be critical to students' academic success and to their eventual sustainable contributions. We view the creation of new assessment methods and practices as a systemic problem, and base our analysis on evidence that the educational system adapts itself to foster the development of the cognitive traits that its tests are designed to measure (Frederiksen & Collins, 1990). A systemically valid test is one that induces the curricular and instructional changes required for an educational system to effectively achieve its valued learning goals. Systemically valid tests foster the learning of knowledge and skills that they are designed to measure. We report here about a project that has been seeking to develop a prototype assessment system that grows out of this approach to systemically valid testing, taking advantage of the capabilities of technologies to include new aspects of student performance in an assessment system. This research and development project is being done in collaboration with a high school to test out these ideas in, first, the domain of physical science.

Two stories illustrate how assessment procedures can have unintended consequences for instruction. A geometry teacher in Rochester, New York was reputed to be one of the best teachers in the state because his students did very well on the Regents geometry exam. But it turned out that he had his students memorize the twelve proofs that might be on the exam--a complete perversion of the goal of deep learning of geometry (Alan Schoenfeld, in press). The second story concerns science assessment. A statewide test on density was administered in Connecticut to eighth and twelfth graders (Sig Abeles & Joan Baron, personal communication). Density is taught in the eighth grade in this state. Students did quite well on the multiple-choice test item where they were given weight and volume and asked to figure out the density. But when they were given a block of wood, a ruler, and a scale, only about 3% of the eighth graders and 12% of the twelfth graders could solve the problem. Students are often learning to give back answers to written items they have no ability to apply in real life.

Our work is based on a theoretical framework which views assessment as a critical and sensitive component of a dynamic educational system. According to this framework, standards for systemically valid assessment include: (1) Directness--in direct tests, the cognitive skill that is of interest is directly evaluated as it is expressed in the performance of some extended task, like the coherence of an argument in a legal brief. Often, directness is sacrificed for "objectivity"; (2) Scope--the test should cover all the knowledge, skills and strategies required to do the complex activity. If part is omitted, students will misdirect their learning to maximize scores on tests, and the featured knowledge or skills will receive disproportinate attention; (3) Reliability--we believe that the most effective way to obtain reliable scoring is to use primary trait scoring adapted from the evaluation of writing to other subject areas. Fairness is critical to any assessment; and, (4) Transparency--the criteria according to which the students are judged must be clear to them if a test is to be successful in motivating and directing learning. (Frederiksen & Collins, 1990).

We have a particular perspective on the kinds of circumstances required for effective education. This perspective grows out of cognitive science and cognitive research, combined with three-quarters of a century of experience with the inquiry-driven student-centered practice that underlies Bank Street's philosophy of teacher and student education. The way in which any new assessment system is designed to structure tasks for students, and to judge their performances and progress should grow out of research on the development of expertise in domains, and the experience of expert practitioners in judging the qualities of students' complex work. The system must be practical on a national scale.

An assessment system that is faithful to the broad scope of learning outcomes known to be critical must assess other processes and performances than those that can be measured by paper and pencil tests. For example, some critical skills and abilities that cannot be so measured include how well students listen and ask questions, how they handle new facts in their theories and questions about consistency, how well they formulate and test hypotheses, how well they make oral presentations and arguments that may make use of visual and graphic media as well as verbal, and so forth. Video and computer technologies also make it possible to construct assessments that are fairer to those students who have not cultivated the test taking skills of the pencil and paper medium.

Technologies add an important new capacity to an assessment system for evaluating these aspects of students' performances. They make new "slices" of performance available to the lens of assessment. Computer technologies contribute a critical new capacity. They have the advantage of making thinking processes available as a student's thinking evolves over time. For example, we are using a modelling program called Physics Explorer to help students develop explanations and theories about physical phenomena. In one task, students use the program to systematically manipulate a set of variables to discover how they affect the period in Galileo's pendulum problem. Their thinking processes as they work on this problem are recorded and made accessible through the program. Video technology allows us to look at the interactive capabilities of people. For example, the quality of students' verbal explanations to each other can be recorded and judged as part of the assessment of their progress.

Components of an assessment system

The development of a new assessment system requires several components. First a set of candidate tasks are needed that reflect the directness and scope criteria. They need to be authentic activities that embody the kinds of knowledge and skills expected of students (Collins, Brown & Duguid, 1989; Wiggins, 1989). Second, a system of scoring students' products or performances needs to be created. In our work, we have been adapting primary trait scoring techniques to science/mathematics assessments. Students' performances are evaluated in terms of a number of primary traits that are clear and understandable to teachers and to students. They should be small in number so that students can focus on them. They should be learnable so that student efforts lead to improvement, and they should cover all the key aspects of good performance on the task.

Third, a library of exemplars is required in order to develop a scoring system and insure its reliability and that it is learnable by judges and students. These exemplars should include critiques by master assessors in terms of the primary traits and they should be available to everyone. This library is key to training assessors and to determine if the system is usable. Fourth, a training system for scoring is needed. There are three groups who must learn to reliably assess test performance: master assessors; teachers; students. Master assessors have the responsibility of maintaining standards, and can work with teachers to help students to perform well.

A prototype assessment system in physical science

The development of the assessment prototype thus must in part be based on actual student performances and products on candidate tasks in schools. The first phases of the project required setting up collaborative conditions in a school that was hospitable to this research--a school where the curriculum and activities well-supported the complex thinking and learning outcomes we sought to measure.

We have begun to work on this set of problems in collaboration with a public high school in New York City--Central Park East Secondary School in East Harlem (CPESS). This is an unusual school, structured around curriculum and scheduling decisions that insure students' deep engagement in complex problems, learning and applying domain knowledge in the context of comprehensive and creative projects called exhibitions. The school has a population of about 500 predominantly minority students spanning the grades from 7 to 12.

During the last year, we have undertaken work on a prototype assessment system in high school physical science/mathematics. We have completed initial tasks in determining the feasibilty and shape of a performance-based assessment system that adapts techniques of primary trait scoring: designing candidate assessment tasks and situations that reveal complex cognitive performances; testing and refining exemplar tasks; collecting a variety of records of student processes and products in different media (including computer-based and video records); constructing initial procedures for judging student performances in physics/mathematics.

At CPESS, the school year is divided into trimesters, the curriculum into two interdisciplinary areas (math/science and humanities), and the school day into significant blocks of time--two hours per class. By the end of the trimester, each student has completed an exhibition that fulfills the requirements for subject-matter learning, and has done an oral presentation for a teacher in which s/he explains her project and demonstrates her understanding of the fundamental science and mathematical ideas. Much of students' time is spent on their exhibitions, producing various forms of written and graphic records. The oral presentation and responses to teachers' queries about their work are central to judgments about student progress. Since students are responsible for displaying competence in several media, our efforts to bring technology in the assessment fit well with the way students' were already judged in the school.

The school's primary goals--the high level thinking and learning skills now prominant on the national agenda for educational change--are ill-measured by our current assessment apparatus. In our collaborative project, the staff of CPESS is seeking to develop new methods for analyzing, understanding and demonstrating student progress for their different audiences (local and national accountibility requirements, colleges and employers, parents). CPESS is thus a natural partner in the research enterprise to develop new assessment methods.

The 9th/10th grade classes are all studying physics/mathematics, amounting to approximately 120 students with a wide range of backgrounds and abilities. It is one of the few high schools in the country where every student studies physics. In the first trimester, for example, students designed amusement park rides and specified--through models, graphs, calculations, drawings, written explanations--the physical motion principles exhibited by their designs.

The technology-based assessment work began in earnest in the second trimester. The tasks that students carried out as they worked on their exhibitions were modified to include the technology, and various records of students' work processes and performances were collected.
The curriculum focused on the physical science concepts in projectile motion. Each student selected a projectile (e.g. a foul shot in basketball, a baseball curveball). Candidate assessment tasks were designed that integrated Physics Explorer into students' exhibition-based work. Students also used Hypercard to fulfill the required model-building aspect of their presentation, and a word processor

and graphing program to present their explanations and display their finished project. Their work-processes and completed exhibition "products" were collected in three different media: computer-based records; paper and pencil; video.

Our introduction of technology into the classroom took two forms. Computer technology was primarily introduced into the physics curriculum through a dynamic modelling program for motion phenomena which enables students to explore and represent a variety of physical phenomena in simulated worlds (Physics Explorer, using Macintosh machines). The program consists of a number of physics models: one body and two body motion, waves, batteries and bulbs, electrostatics, and so forth. Students can set various parameters associated with each physical model to control its behavior and observe the effects. They can plot parameters against each other in graphs and type ideas or explanations into a note-taking space. This software enabled us to structure tasks for students that, for example, asked them to hypothesize about and test the effects of friction on physical motion variables, and to create records of their tests in the form of graphs and charts. Video technology was introduced to create records of students' interactions with their teachers as they presented and were queried about their work, and to structure tasks that recorded their interactions with each other. By videotaping students' initial presentations of their exhibitions, we collected baseline records to begin to develop a scoring system for students' explanations and answers to difficult questions.

We have tested a task that relies on video technology to capture aspects of students' knowledge through their interactions, students prepare explanations of physics concepts they have studied. Students are then paired. One student explains how, for example, gravity works to the listening student, using a blackboard or other visual aids. The students then reverse roles. The videotaped interactions are judged for qualities like the ability to interact cooperatively, to ask good questions, to adapt the explanation to the listener's needs and so forth.

Scoring procedures

The records of students' complex work that we have collected in different media are serving as the basic data for the development of a scoring system based on primary traits. We want to develop different sets of criteria to evaluate different aspects of students' performances such as explaining, listening, responding to questions.

Different groups of people who have a stake in the assessment of physics/mathematics learning have generated groups of primary traits for different aspects of students' performances (teachers, cognitive scientists, assessment specialists, science educators). For the initial generation of criteria, these experts ranked a set of student products from the collection of projectile motion exhibitions. They then generated the criteria by which they made those judgments. The criteria need to systematically discriminate among performances on the selected aspects of performance. For example, with respect to student explanations, criteria might include clarity, coherence, integration of visual and verbal material, monitoring the listener's ability to follow the explanation. Different levels of performance need to be distinguished for each criterion (we anticipate that there will be four or five levels). This will require finding different levels of performance in the student records, writing descriptions of each, and illustrating them with clear examples and critiques.

These initial lists of criteria were compiled, and sent to a second group of judges along with a set of exemplars of student performances that represent a broad range of quality and different aspects of performance. The task for this second group of judges is to score each of the exemplars according to each of the criteria, attempting to specify what counts as different levels/kinds of performance on each. They also make judgments about which criteria are key aspects of expert performance. The second set of independent ratings will again be compiled to identify which aspects of performance are consistently rated as key, and judgable. This distilled set of primary traits and associated performance levels will be examined for scope, and then applied to different sets of exemplars.

We will teach a new set of judges to use it in order to determine its reliability, and the best methods training assessors. We suspect that videotape will be an important technology for training these discriminations. We plan to train three judges in the system, using the exemplars and critiques. They will then judge all the collected materials in order to determine interrater reliability. Similar records will be collected in a subsequent year to see if the system can be used to reliably judge students' work in the following year. The scoring system must also be tested with students to see if they can use it to improve their own performances.

We anticipate a number of future tasks to expand the scope of this prototype system and more broadly determine its adequacy and systemic effects.
We will extend the research and development from physics/math to other subject matter areas. We will first adapt this assessment approach to biology/math--a near transfer domain. The staff at CPESS is eager to begin work on the assessment procedures to this subject area.

The research will be expanded to include additional schools. A suburban school system has been a collaborator on a number of other projects, and is now eager to undertake joint investigation of this approach to assessment. Work with different schools and populations will provide a more general test of the tasks and scoring procedures, insuring a robust system.

The outcomes of such assessment procedures are complex compilations of information. It is not just the master assessors who need to make judgments about complex performances, but the audiences for the information who need to fully and accurately understand the assessment. We will therefore conduct experiments about how to best represent the outcomes of the assessment procedures for different audiences. There is a large and relatively unexplored problem involved in creating representations that are efficient and understandable to different audiences, yet do justice to the complexity of the student learning we seek to assess.

Negotiated *Topoi*, Networked Epiphanies:
Toward Future Technology Assessment Methods and Madness

Hugh Burns
The University of Texas at Austin

Abstract. The ends of technology assessment depend on how well the measures by which we assess and the measurements by which we know are agreed upon. In the future, constructing the measures and measurements in collaborative communities of designers, developers, and users, perhaps with the assistance of local area networks, will help insure a successful technology assessment. Recent experiments with realtime electronic mail, "groupware," "integrated electronic discourse communities" are providing models reaching consensus. The evaluation community is moving toward more negotiated topics--*topoi* , i.e., agreed upon standards for assessing purpose, act, scene, agent, and agency. Likewise, the technology assessment community will be using new technologies--making networked discoveries or epiphanies. At the University of Texas at Austin, such systems are being used in substantial writing courses across the curriculum. An examination of the communication behaviors of network users in such a course provides a model for using groupware techniques in technology assessment and on local area networks. In discovering collaborative standards and negotiating within realtime assessment communities, the future of technology assessment means (1) more reliable methods for reaching consensus and (2) more valid instruments for predicting individual efficiencies and organizational effectiveness.

The fundamental argument of this paper is that *the ends* of a significant and successful technology assessment depend on how well the measures by which we assess and the measurements by which we know are agreed upon. The fundamental prediction here concerns *a means* for insuring significant and successful technology assessments. Constructing measures and measurements in collaborative communities of designers, developers, and users, with local area network technology, appears most promising. The evaluation community is moving toward more negotiated topics or *topoi*--agreed upon standards for assessing purpose, act, scene, agent, and agency. Likewise, the technology assessment community will be using new technologies--making networked discoveries or epiphanies.

Toward this end, this paper examines two "estimates of the future" and offers an illustrative, annotated case example. These estimates are: (1) technology assessment methods will depend more on recovering topics, *topoi*, through collaboration and negotiation, and (2) technology assessors will use local area networks and establish their own electronic discovery community. In a case study, students in a substantial writing component course at the University of Texas at Austin deliberated on-line about how computers upset the traditional scenario--how computers, in the words of Sherry Turkle, "upset the distinction between things and people" (61).

Toward Negotiated *Topoi*

Topoi--places where one constructs knowledge--provide significant insights for exploring the problem of estimating the future of technology assessment and engaging the dynamics of assessment. Using "topoi" allows a more systematic inquiry. Whether informing, persuading, expressing, narrating, describing, classifying, or evaluating, *topoi* help clarify the right questions (Aristotle, 1954; Kinneavy, 1971; D'Angelo, 1975; Burns and Culp, 1980). Technology assessment certainly requires a systematic construction of measures and measurements, or, in plain language, a way of describing and evaluating how the technology works and how well it works. Today, this assessment process means more knowledge acquisition and knowledge representation. But a theory of topics will not necessarily be new. Technology assessment, it seems to me, will recover through collaboration its own *topoi*.

Recovery: Topoi for Identification. One of the most powerful heuristics for generating the "right" questions is attributed to Kenneth Burke. Burke's "dramatistic pentad" is fully described in *A Grammar of Motives* (1969). The five key terms of dramatism--purpose, act, scene, agent, and agency--represent specific perspectives humans share in attributing motives (xv). Specifically, Burke contends that "any complete statement about motives will offer *some kind of* answers to these five questions: what was done (act), when or where it was done (scene), who did it (agent), how he did it (agency), and why (purpose)" (xv). In technology assessment matters, these questions help recover what it is known. They help identify issues.

But using dramatism only to identify *who, what, when, where,* and *why* issues does not exploit the potential power of this methodology. The potential complexity of an inquiry using the correlations, associations, and combinations of these elements Burke termed "ratios." The exploratory appeal of ratios may be attributed to Aristotle's classification of causes, for Burke specifically traces the pentad's evolution through both Aristotle and Aquinas:

> The most convenient place I know for directly observing the essentially dramatistic nature of both Aristotle and Aquinas is Aquinas' comments on Aristotle's four causes. . . . In the opening citation from Aristotle, you will observe that the "material" cause, "that from which (as immanent material) a thing comes into being, e.g. the bronze of the statue and the silver of the dish," would correspond fairly closely to our term, *scene*. Corresponding to *agent* we have "efficient" cause: the initial origin of change or rest; e.g., the adviser is the cause of the child, and in general the agent the cause of the deed." "Final" cause, "the end, i.e. that for the sake of which a thing is," is obviously our *purpose*.. "Formal" cause ("the form or pattern, i.e. the formula of essence") is the equivalent of our term *act*. . . . We can think of a thing not simply as existing, but rather as "taking form," or as the record of an act which gave it form. . . . (228)

A theoretical "topical" scheme for estimating the future of technology assessment thus begins here--an arrangement of questions concerning purpose, act, scene, agent, and agency as well as their twenty ratios, i.e., purpose/act, purpose/scene, purpose/agent, etc.

Negotiating "Purpose". Where does technology assessment begin? Why should we assess technology? What are the goals of technology assessment? By agreeing on the purposes of an assessment, a community better understands what may be accomplished. The first law of a new technology assessment methodology should begin with a negotiated sense of the purpose or function of a technology. So let us begin with Lewis Carroll's *Alice in Wonderland* since assessment depends, as Alice learns, heavily on mutual communication and negotiated understanding.

> The Mock Turtle said. "No wise fish would go anywhere without a porpoise."
> "Wouldn't it, really?" said Alice, in a tone of great surprise.
> "Of course not," said the Mock Turtle. "Why, if a fish came to *me*, and told me he was going on a journey, I should say, 'With what porpoise?'"
> "Don't you mean 'purpose'?" said Alice.
> "I mean what I say," the Mock Turtle replied, in an offended tone (97).

Although Alice will be unable to negotiate a definition of purpose with the Mock Turtle, the technology assessment community should work as a community of purpose, on purpose. What are the topics for identifying or recovering purpose? Three come quickly to mind: consequences, conditions, and experiences.

Consequences. Especially in this age of everchanging technologies, ambiguous human-machine interactions, and evercharged political and economic contexts, technology assessment should describe at least two kinds of consequences: (1) what sets of activities are empowered by technology, (2) what sets of activities are endangered by technology. Technology assessment allows users to know what the technology is good for, what differences it will make, and whether or not it works satisfactorily. A pragmatist calls this utility, but the technology should be tested to see how faithfully it satisfies the desired agenda.

Conditions. Are constraints imposed because of the desired purpose? Are the conditions under which change occurs appropriate considering the quantity of the technological change or the quality of the technological change? Are there speculative methods which can be developed before the technology has its own history?

Experiences. Experience always has been a great and grand teacher when it comes to assessing new technologies, new ways to behave and to "improve" performance. Comprehending both consequences and conditions is much easier if actual experiences with a technology are available. Special experiences are especially useful since they provide a personal context for understanding whether or not a technology is desired, needed, or essential. Not surprisingly, when Alice lets the Gryphon and Mock Turtle know that she has not experienced life "under the sea," they explain to her the way it is--where education is more about "Reeling and Writhing" than reading and writing, where the functions of arithmetic are "Ambition, Detraction, Uglification, and Derision." What is the lesson of negotiating purpose--one we technology assessors may need to be reminded of as we approach the uncertain future? Simple. Have one. If technology assessment methods lack purpose, then where will the standards originate? With gryphons and mock turtles, that's where.

<u>Negotiating "Act".</u> What is happening? What is the form of the technology being assessed? What are the topics for identifying act? Design tendency or a technology's disposition, a dynamic communication capability, and the essential explanatory power are among the *topoi* for negotiating "act."

Disposition. Technology has disposition--features in its design which account for its function. These dispositions should be apparent if not transparent. The relationship between purpose and act should be evident in the design. While it may sound like a pathetic fallacy, a technology's disposition or tendency to function in a particular manner allows humans to extend their imaginative reach. Technology is never content for long, with either its form or its function. In fact, as this disposition is "actualized," a technology reveals not only itself but also its next incarnation. The first moments of this "action" process are critical. In such moments, the disposition of an event can be predicted from the form alone. Immature technologies are suspenseful and surprising because of their incompleteness, for technologies are seldom finished or complete. The recognition of disposition in a mature technology makes assessments easier.

Dynamic Communication Capabilities. The aim of an assessment obviously depends on how the overall communication is conceptualized--the more revealing the better. The newest generation of computer software will be explored more as a dynamic medium in its own right, vastly different from books, film, and television. For example, the advantages and disadvantages of a dynamic text will force contemporary scholarship to be deeply concerned with texts and objects as a forms of knowledge representation and, thereby, with investigating the nature of discourse processing itself. Fresh concepts such as hypertext and hypermedia will emerge as most important attributes of an evolving set of advanced technologies as Edward Barrett's Text, Context, and Hypertext (1988) and *The Society of Text* (1989) illustrate. Shoshana Zuboff's In the Age of the Smart Machine: The Future of Work and Power (1988) also presents many dynamics of transforming information models to knowledge models--in the human mind as well as in the workplace. Tomorrow's communication software should be designed to allow users to think and perhaps even feel the consequences of their various choices. As such hardware and software matures and as such computing power is made widely available, evaluators will find platforms for investigating how humans learn to use technology.

Explanation and Essentiality. From a developer's viewpoint, generating explanations is an essential goal in both design and assessment. How are explanations driven by the performances? In the near-term, computers will provide recordings of problem-solving processes so that researchers may observe and review the actual interactions and evaluate how an expert performance occurs. We have all had the experience of trying to understand something only to realize that--while we thought we understood the problem--we were unable to articulate what we knew exactly or what we understood as "facts of the matter." Explanation knowledge can also be used to develop methods to generate answers and further elaborations. Describing what we know in words or figures so that a computer program can present that knowledge effectively is the complex science, perhaps art, of generating explanations. The knowledge in our heads and hearts is not always the precise knowledge on the page or in the data base. The essence of an explanation is obviously within the user. The issue becomes how to design computer-assisted tools which are learnable because they are self-revealing. Future systems promise to be even more helpful and more intelligently active.

Negotiating "Scene". Understanding the settings for technology assessment should not be ignored. In fact, scene issues are often the most complex. If done well, then technology assessment initiates new designs and new developmental agendas with specific situations in mind. Perhaps the future promises those of us in evaluation, testing, education and training technology some of the very same wise insights. What are two topics for identifying scene? Accommodation and propriety.

Political and Cultural Accommodations. How to establish a common ground in the assessment setting to account for the political, social, and cultural variables is always problematic.. The nature of knowledge today depends heavily on what happens within the political setting. These settings actually determine whether the knowledge is constructed or deconstructed, used or abused, shared or not shared. Validation. the stability of knowledge in community, as well as many of the integrating issues and public considerations will depend on aspects of cultures and levels of power.

Propriety. A coherent viewpoint must also be tailored to an individual user's needs. Proposed architectures for an appropriate implementation can be described both in terms of how that knowledge is understood and how it can be misrepresented by users and experts alike. The consideration of being able to judge what is the "right" technology for the specific situation makes technology assessment training so very important. What we have learned by bureaucratic experience is that our sophisticated planning and sophisticated instrumentation is not enough if the ability to understand what is appropriate is not well understood or agreed upon.

Negotiating "Agent". The people who assess and are assessed help focus another set of topics. Respect and readiness are two of several topics for negotiating matters of agent.

Respect. How does one privilege user's observations through negotiation strategies? In the context of transforming information models to knowledge models, such radical new paradigms invite more in the way of respect for ideas and tentative hypotheses. Sometimes in a network, respect is discovered in surprising ways: in monolog (writer-to-self), dialog (writer-to-one other), and polylog (writer-to-many-others).

Readiness. One of the most obvious advantages of tomorrow's technology implementation will be simply having more prepared people. More general literacy and more scientific literacy will be necessary if we are to survive as a society. Anyone who is trying to improve performance of a skill can attest that the more time they can spend in practice, the greater the likelihood that a performance will improve. Preparation tomorrow will require a "compression of process" which should allow habits to be reinforced more fully.

Negotiating "Agency". Of course, the tools themselves must be assessed. The instruments must generate their own set of topics. If done well, then technology assessment initiates new designs and new developmental agendas. Perhaps the future promises those of us in evaluation, testing, education and training technology some "intelligent" instruments. Because advances in software design have allowed us to capture more performances, researchers should now focus on that data and examine ways to develop more formal specification tools that can be used to describe human behavior, independent of software implementation. What are the topics for identifying topics of agency? Design elegance and flexibility are foremost.

Design Elegance. Develop elegant and simple tools so that even people who are not as familiar with computers can immediately start working. Another practical payoff might be the capability to design simultaneously a technology assessment toolkit for the communities of advanced technology users, designers, and developers. The impact of such technologies is only beginning to be discussed and interpreted; studies such as Terry Winograd's and Fernando Flores's Understanding Computers and Cognition: A New Foundation for Design (1986) are stimulating the discussion of future design dynamics.

Flexibility. Many feel that today's inflexible or brittle software does not significantly help them meet their technology assessment needs. Tomorrow's answers will result from flexible software that permits more opportunities for realtime communication as well as community-controlled modifications for on-line conferences and discussions, electronic mail for extending the communication, and off-line evaluations of evaluations and evaluators. In the long-term, as more and more "artificial intelligence" is designed into systems, applications then may be even more individualized. These trends are unmistakable.

Estimating Patterns, not Practices. While many more uncertain implications than certain conclusions are likely in the near-term, recovering a theory from "topoi" and discovering the social patterns and practices of networked communication are certain to influence the precision and believability of future technology assessment. Such understandings will allow more reliable methods for reaching consensus and, thereby, more valid instruments for predicting efficiencies and effectiveness. Not what about discovery?

Toward Networked Epiphanies

While this theory recovers knowledge, recovery alone is not enough. A theory of technology assessment needs an additional multiplier in order to discover new knowledge. Technology assessment need an aesthetic theory in order to appreciate qualitative differences. Such a multiplier may be found in the notions of "epiphany," of insight, and of illumination. "Making meaning" means constructing an foundation for understanding one another, but true meaning should also ultimately serve beauty. Yes, truth is beauty, beauty is truth.

While technology assessment evolves more toward a social construction of meaning, the future of technology assessment is bright simply because the future could be quite "beautiful." Collaborative technology assessment instruments will allow realtime evaluation, equitable access to information, as well as the potential for revolutionary understandings about learning. This revolutionary understanding will come from applying the standards of epiphany.

To realize discovery, each of the primary motives--purpose, act, scene, agent, and agency--must be examined for three additional qualities or conditions of beauty: (1) integrity or wholeness, (2) harmony or symmetry, and (3) radiance.

Discovery: Topoi for Creative Insight. *Integritas, consonantia, claritas*--understandings of integrity, harmony, and radiance are well-seasoned principles for achieving new insights within the entire technology community. When Thomas Aquinas originated this particular line of aesthetic inquiry, he was speculating on the discovery of a soul. Likewise, an evaluation ought to be searching for the soul of the matter. In his *Stephen Hero*, James Joyce writes of these Thomistic ideas: "First we recognise that the object is *one* integral thing, then we recognise that it is an organised composite structure, a *thing* in fact: finally, when the relation of the parts is exquisite, when the parts are adjusted to a special point, we recognise that it is *that* thing which it is. Its soul, its whatness, leaps to us from the vestment of its appearance. . . . The object achieves its epiphany. . ." (289).

Toward Networked Integrity. In a 1975 article in *The Futurist*, Jacques Vallee asserted "A scant 100 persons throughout the world now use computerized conferencing on a regular basis. But the time may be fast approaching when far more people will be conferring through computers and we will begin to view computer conferencing as a 'natural' way to interact " (298). The "fast approaching" prediction was absolutely true. They also used terms such as "invisible colleges" (298), "fast thinking that would enhance our collective abilities to resolve conflicts, deal with crises, or improve decision-making capability" (299). They also pointed out how useful "transcripts would be as threads or chains of thought were created and later evaluated" (294). All true. Now these truths have become integral. We speak of a network as a single element--whole, but not necessarily indivisible.

Certainly, collaborative communication media would attract even more attention. The times are fast approaching when far more technology assessors will be conferring through computers. They will begin to view computer conferencing as a 'natural' way to interact. Also fast approaching will be local area networked classrooms which will focus attention on fundamental skills such as writing well, reading well, and thinking critically. The "invisible colleges" will be anything but invisible as such environments multiply as you shall soon see in the following case analysis. Such tools for widespread communication will enliven the topics of an assessment.

Toward Networked Harmony. Realtime communication among teachers and students is providing transcripts and artifacts for this investigation. In the interactivity, one sees the parts. Truth seems more tentative, more relaxed. The certainty factors change. "Interactivity" is the real strength of and hope for computers in assessment settings. The set of communication tools and associated activities in a local network configuration provides a mini-forum for investigating, exploring, and stimulating the "knowing" processes. LAN software capable of helping assessors experience and manage complex problem-solving skills required for extremely flexible protocols is being developed. The issues surrounding the design, implementation, and the evaluation of flexible communication environments are being refined: e.g., defining user/computer interactions. The design of the systems is moving toward learner-centered, reactive learning environments. LAN solutions may help such users by working collaboratively, thus being able to negotiate the meaning and the construction of the parts together.

Toward Networked Radiance. Radiance is possible only when people are motivated to keep imagining and to keep at the construction of knowledge. Extensions to network capabilities are critical to realizing this potential. For example, within a local area network environment, a netmanager allows other assessment tools to be accessed. Radiant categories could include editors, message systems, and utilities. A practical radiance can be seen in pull-down menus which reveal word processing tools, spreadsheet systems, organizers, etc. Message systems can be expanded to include organizers, "smart" in-boxes, task assignments, electronic mail options locally and globally, and other conference-enabling communication tools. Users can look at yesterday's discussions as source documents for assessment. Networks become real tools for knowledge sharing rather than just electronic conversations, thus becoming rather than being an assessment.

The Case

The following transcript is part of the data being collected in research at the University of Texas at Austin in the English Department's Computer Research Laboratory. The general research investigates issues in integrating writing software in substantial writing courses. In the lab, various methodologies for designing realtime classroom environments are also explored. In this setting, technology assessment tools and techniques are being developed for documenting an "electronic discourse community." Specifically, the research team has designed and developed tools and software features which can be assessed in delivering writing instruction on a local area network (Bump, 1990). "Interchange" developed by The Daedalus Group integrates collaborative, groupware functionality in a system which allows for a full textual interaction. Interchange is a realtime tool in the integrated system.

Speculating from recent experiments with an "integrated electronic discourse community," technology assessment also could benefit from exploiting the social dimensions of evaluation. In plain language, how people communicate and collaborate on networks will be useful for coming to terms with measures and measurements. Therefore, the communication behaviors of network users in this particular intensive writing course provide some examples of new ways to communicate and to learn. By analogy, such "writing-intensive" case models have implications for technology assessment tools in the future. This case presents a rationale for recovering the practices we practice as well as setting an agenda for conducting "technology assessment" sessions in the future.

Looking to the transcript, the numerical notation within the brackets refers to the number of the message in the chronological sequence of messages, the specific number of an individual sender's message, and the message which is being responded so that the hypertext can be traced. With the exception of my name, all of the participant's names have been changed. Let me acknowledge my students in my "Introduction to Computers in the Humanities" course. I deeply appreciate their willingness to explore these evolving tools, not to mention their ready wit and wisdom.

```
Hugh [1:1:0]:
Okay, just how do computers upset the traditional
scenario? See Turkle's description on page 61. I
thought she was making an excellent point about the
physics of computing versus the psychology of
computing. Let's start there and see what you all
thought about her approach. By the way class, I have
invited Jacob Welton to share these INTERCHANGE
moments with us. Jacob teaches at The University of
Western Cape in Capetown, South Africa. He has been
helping me understand more about Black literature,
multiculturalism, and especially to many writers I
have not read: Achebe, Coetzee, etc. Welcome, Jacob.
```

I began the Interchange session by announcing my purpose of the session: to discuss one of Sherry Turkle's major themes in *The*

Second Self. Turkle writes: "Computers upset the traditional scenario. First, they upset the distinction between things and people; it can no longer be simply the physical as opposed to the psychological. The computer too seems to have a psychology--it is a thing that is not quite a thing, a mind that is not quite a mind. And they upset the way children perceive their 'nearest neighbors.'" (61) Turkle is making an important point, and I want to make certain that all my students have thought much about these physical/psychological distinctions. This is the purpose which will be negotiated as you will see. In terms of foreshadowing how the class will respond in this environment, I also mention that we will be interested in watching how Turkle works, her methodology. Finally, I welcome Jacob Welton, a visiting scholar from the University of Western Cape. One of the advantages of the local area network is inviting others to participate on the spur of the moment. In summary, this discussion opens with a statement of purpose, a prompt about a researcher's agency, and an elaboration of the extended scene with the presence of a guest. Natalie responds first to the scene by calling this interchange babble--a comfortable and often appropriate description of what happens; Vincent and Marilynn begin to negotiate the purpose.

```
Natalie [2:1:1]:
Hello Jacob, I hope you will enjoy our "babble" on
interchange!

Vincent [3:1:1]:
One thought I had while reading....  the "traditional"
relationships that children have made about what is
alive, thinking, etc. comes from the lips of adults.
Nothing is upsetting to the child about the changes in
these relationships due to the computer and its
broader mind.  The child learns what is around.  What
kind of questions should the adults be asking?  "Is it
alive?" seems kinda strange.

Marilynn [4:1:1]:
Children's notions about their first encounter with
computers upsets their traditional scenario that
distinguishes alive from not alive.  However, perhaps
this is a fitting orientation to the modern artificial
intelligent world in which they will grow up.  The
Merlins and the mutant speak and spells serve to
```

initiate young children into the world of machines
that will soon be psychoanalyzing their users.

Vincent sees the issue as "kinda strange." Marilynn finds the orientation "fitting" in the "modern artificial intelligent world." I enter my next message, fulfilling my earlier promise to have the discussion also consider Turkle's methodology. Here, then, we have two topoi--one of purpose and one of agency. The discussion thus has two edges. While I am posting this message, our guest, Jacob, has already figured out the editor and now introduces himself to the electronic audience. We are all in the same room, but on the network we are what we write--faceless, but not voiceless. Jacob indirectly introduces yet another topoi--the differences between technology in the Third world and the First world. Miranda joins the conference, pointing out two reactions.

Hugh [5:2:0]:
Turkle: (p. 29) "My style of inquiry here is enthnographic. My goal: to study computer cultures by living within them, participating in their lives and rituals, and by interviewing people who could help me understand things from the inside." I think we all could be subjects for her next book, *The Third Self*. We have been living in this culture for five weeks or more, what is alive in here? What is ritualistic here? What would Turkle say about us?

Jacob [6:1:1]:
Many thanks, Dr. Burns (Hugh). I feel as if I have entered another time zone here. The technological wizardry of the First World is strange to those of us from the Third, but the feeling is not too unpleasant!

Miranda [7:1:1]:
Indeed it does seem that computers "upset the traditional scenario" by confounding the distinction between things and people. It is interesting to me that there are two sort of responses to this: 1) People try to emulate the perfect rationalism of the computer, striving to make their thinking more neat and, in contrast, 2) People gain a new respect for what the computer can't do, i.e. it can't have emotions or values.

Hugh [8:3:6]:
Jacob, how is such technology perceived in the Third world?

Peter [9:1:1/3]:
Turkle's got something here. Her grappling with the realness of computers is best shown when she describes its "marginalized" place in the human world. Like the spider, it is kinda alive, but not fully alive. As she states, "...a culture is in the process of growing up around computers that surrounds them with a discourse of almost-life." (p. 59)

Miranda [10:2:5]:
I like Turkle's methodology, i.e. ethnography. It is clear that she has immersed herself in her topic. I thought she was somewhat redundant, but her thoughts and observations were fascinating to me!

Natalie [11:2:1]:
What I found the most striking was that computers seem to occupy a position somewhere in between human beings and non-human beings in children's minds. On the one hand they are assigned distinctly human qualities like emotions or intelligence; on the other children seem to be aware that these qualities have been "put inside" a machine. It made me think of how I sometimes have to remind myself that computers are only machines when I started to perceive them as evil fiends out there to make my life harder.

Vincent [12:2:6/1]:
Greetings, Jacob. I find it hard to talk about a computer culture. I'm not sure I understand what that is, it seems that computers have not been around long enough...its hard to point to things that are special to it. I can see the roots of techno expectations going way back to the first digi-gadgits. If there is a culture to get inside of, is it more that the expectation of the new and better and more alive?

Ariel [13:1:4]:
What is going to happen when the children start using much more advanced AI-type computer toys as opposed to the simple ones like MERLIN and SIMON? The line between alive and not alive is going to be fuzzier.

If there is a pattern emerging in the discussion, the participants are generally responding the "traditional scenario" in their first response and to Turkle's methodology in their second. But the neat distinctions soon disappear as the community begins to respond to each other. Ariel answers Marilynn [4] and asks a question about the consequences of even more advanced AI systems, concluding that the issue will be even "fuzzier" in the future. Now Carlos and Tenley arrive together. Now the class is complete. Everybody has posted their first message, and we reach the 100% participation level. Not bad for a class discussion, and so hard to do in a "real" discussion. Tenley pays particular attention to the agent issues--noticing how adults attend more often to "live/non-live" distinctions and her quick feedback for Carlos. The reference to the portfolio [17] is one of the class assignments in which each of the students prepares a commentary essay based on class notes, these interchanges, readings, and the electronic mail. As a matter of fact in my next message [18], I use part of Miranda's previous portfolio, a message taken from a previous interchange session. The file convention is ETPMIR.376 and indexes the "Electronic Text Portfolio from MIRanda in English 376." In other words, all members of the class have the capability to use any previous discussion since all of the class sessions are stored in the course's emerging electronic library--the exploitation of conversation. Miranda reminded the class of some of the gushy ideas I had earlier about the nature of an electronic community. She asks a question about Turkle's methodology--a question assessing the tool itself. Meanwhile, Ariel enters a message asking "anyone" to help her understand Turkle's conclusions and to verify that she is on the right track--that the bottom line is an issue of control and power. Ariel [19] loves asking questions. Questions, of course, are topoi--new places from which we may reach networked epiphanies.

```
Carlos [14:1:1]:
I think children are an excellent source for
observation when it comes to the study of psychology.
The concept of "infinity" and whether things are alive
or dead according to motion (Piaget's studies) as
children perceive the world, opens a sense of
direction toward the psychology of computers.  Turkle
argues that children see a computer as a psychological
entity and not a physical one...this is what I mean by
such studies giving a sense of "direction" to us, the
adults, readers, as we sit and contemplate the degree
of intelligence of computers.

Miranda [15:3:1]:
```

I liked her reference to animals. Children used to define humans in the way that they were unlike animals, i.e. smarter. Now, it appears that people may not be smarter, but have emotions. And, curiously, it has always seemed that animals -- especially dogs -- are possessed of pure emotion. Which is why people so often become so attached to dogs, and in a way which they could never become attached to a computer?

Tenley [16:1:1/7]:
Two comments: Kids are so imaginative and un-linked to adult reality that as somebody said, it is probably adults who think the live/non-live issue may confuse kids. The kids probably have no problem with it. Secondly, I think Miranda is right when she said, as we've discussed before, that we try to be more like computers and conversely discover their limits.

Tenley [17:1:14]:
Wow! Carlos, good point. I will no doubt have to use that quote about Piaget in my next portfolio.

Hugh [18:4:5/10]:
I have brought this observation in from one of your files on the network to illustrate that what you all have said is not forgotten. Here is something that makes me think I think we are alive. I clipped this from ETPMIR.376: "And another word from our sponsor: (Hugh Burns) 'This is a community, and the more that we achieve COMMUNITY here, the better the class will be. Believe me, I do not do this for the big bucks. I do it because I am absolutely amazed at the energy of this kind of classroom and these kind of features. I will be watching the back of your heads but listening to your voices and even your hearts.'" What would Turkle make of this in an ethnographical kind of way?

Ariel [19:2:1]:
I'm not sure I understood what her conclusions were for the example of children wanting to "kill" and "resuscitate" the programs. Is it because they like the control? Anyone?

Vincent agrees with Peter [9] who has speculated that Turkle may stretch the continuum too far. Vincent agrees that the matter may be "more personal." Miranda [21] begins a self-referential moment; she reflects on her own message [15], again asking a question and establishing yet another new topoi: <u>if</u> kids with technology, <u>then</u> what will the adults be. With the "electronic village," Tenley echoes Toffler, McLuhan: cascading images of third-wave media, message, meaning, method, madness. Our visitor from South Africa, Jacob, has been composing an answer to how technologies are perceived in the Third World [8] by imagining how his students at the University of Western Cape would respond to such a class. Marilynn comments on batteries, while Miranda "wonders" about the imagery of infinity; Marilynn concentrating on the machine, Miranda concentrating on the memory. Miranda's "topoi" on pure emotion [15], kinds of adults [21], and infinity [25] will prompt another set of comments for Tenley, Ariel, and Natalie. Tenley [26] wonders about producing "super kids." Ariel [27] reminds all of us how we have all become lost in the mirrors in the mirrors in the mirrors. Natalie [28] challenges Miranda's assumption and wants proof for the assertion "aliveness" concepts.

```
Vincent [20:3:9]:
Peter--I think I agree with the notion of a discourse
of almost-life.  I think that's the way that a
"computer culture" will see more of the world.  It
does put an emphasis on what we consider most lifelike
in ourselves, and does not necessarily have to be
emotional/physical dichotomy.  I think it will be more
personal.

Miranda [21:4;15]:
What I meant in my last fuzzy message was that people
may not be able to think of themselves as smarter than
computers (although we still think -- probably
correctly -- that we are smarter than animals).  I
think it is fascinating to think that children are
spurred by this new technology to be reflecting on
psychological concepts at such an early age.  This
parallels Seymour Papert's observation that through
computer graphics kids could comprehend such advanced
concepts as weightlessness at an earlier age.  If
children are grappling with all these concepts at
early ages, what kinds of adults will they become?  I
tend to be optimistic about this, although the
generation gaps may become ever broader!

Tenley [22:3:5]:
```

I think we are in an electronic village, an electronic community. Obviously, our speech acts are different, but there is an ethnography to it.

Jacob [23:2:8]:
Dear Hugh, you asked me how computers are viewed in the third world. I notice the term ethnography cropping up here. I think from the perspective of a technologically less advanced society, computers mean power. Not power to control and order the world you are in but power which is beyond one's control. On the other hand, I can see that if my students from the townships of Cape Town were to participate in an environment such as this one, the response would be a feeling of gaining in humanity, by being part of a communicative network in which power is demystified.

Marilynn [24:2:1]:
I especially found it amusing when the kids thought they could "kill" the machine by ripping out the batteries. Then, after other children viewed the computer killing process, they also wanted a chance to kill it, then bring it back to life. I agree with Turkle that this lessens the mystery of the machine that appears to be alive when it scarcely will not respond to the off button. Although children probably enjoy this process of killing it just when it starts to resemble something living, the knowledge that it can be shut off is certainly reassuring for example with Laura.

Miranda [25:5:21]:
I'm wondering if anyone here was traumatized by the picture on the Quaker Oat Box, Salt Box, etc. which depicted infinity -- as Turkle refers to it on a book cover. I do remember staring at this box as a child and being intrigued, but not frightened.

Tenley [26:4:21]:
Miranda et al. It is interesting to think what all of this will mean for the next generation of adults. I do think that working on computers makes one think faster. Sometimes I get frustrated that traditional classes seem to go slow compared to computer-assisted ones. I wonder if we really will produce super kids.

Ariel [27:3:25]:
Miranda, I really go into infinity in two mirrors mirroring each other...

Natalie [28:3:15]:
Miranda, has it really been proven that children start to tackle these psychological questions earlier in

their lives when they are confronted with a computer? I find that somewhat hard to believe. If they are able to contrast the "aliveness" of a spider to that of a computer, obviously they must have had a concept of it before their first encounter with a computer.

Now the "InterChange" is expanding is several directions. Vincent will pick up on Marilynn's batteries as "a life force." People continue remembering their own close encounters with infinity, e.g., Tenley [30]. Miranda decides to converse with our guest, Jacob, but as usual brings another topical dimension to the session: the political consequences of introducing technology in the Third world. What will Jacob say? Meanwhile, Hugh as "teacher" took Ariel's bait [19] and responds to the "Anyone?" ploy, but obviously could not locate Turkle's exact point. Nevertheless, he speculates on the power as an ability to keep others from not knowing: yes, another topoi.

Vincent [29:4:24]:
In response to the batteries, I thought Turkle's observations were neat. Batteries are the end to the reasoning. They are food, a life force. To me, that mystification, is a different sort. But it stresses the comparison to life.

Tenley [30:5:25]:
Miranda: I don't remember the quaker oats box, but I can distinctly remember the night I thought about infinity and got really upset because I couldn't grasp it and couldn't stop thinking about it. I was probably about 8.

Miranda [31:6:23]:
Jacob -- It would be great if you could bring computers to third-world citizens. It would be an invitation to join the literary world, which in modern terms as you say, is joining the world of power. I wonder what powers might try to keep computers out of the hands of third worlders for just that very reason . . .

Hugh:
Ariel, what page were you on? I am sure that the kids like the control--humans too. The group I found interesting was the techno-kids who made it seem that the computer had crashed when all the time they had only made it seem to die. Magic. Presto. The consequences of this kind of kid? Behavior or what?

First, power is knowing how to use the technology. Second, more power is knowing how to have others NOT KNOW how to use the technology. In the second case, the culture creates a DEPENDENCE on the techno-kids? What do we do if the network really crashes and, heaven forbid, we would have to talk to each other face to face. Call Barry the repairman next door. That's control and that's power in this culture. I'll save your AI toys to comment on in a minute--interesting question.

Here comes Barry. Barry, who? Now Barry is not even in the class, not even in the room. Was he watching for me to mention his name. I just did. In any case, Barry is the lab's technical assistant and all-around troubleshooter. He has logged on to the network from the research lab to lurk and to report on the new Hypercard 2.0 manual which I had just purchased. When it comes to electronic networks, *Mi casa, su casa.* I find it hard to imagine that a friend would walk into a classroom discussion and say:

Barry [33:1:?]:
Hugh--Sorry to jump in uninvited, but I just ripped through Goodman's Hypercard 2.0 manual--Jumpin' Jehozaphat! its sooo neat! You can have multiple windows and everything. Looks like no studying tonight.

Ariel also wants more proof from Miranda, so comes to Natalie's rescue. Hugh was not the only one to take Ariel's bait [19]; here comes Carlos. I am struck by the number of images which Carlos repeats in his answer to Ariel. He has used Tenley's ideas [16], Miranda's dogs [15], and Marilynn's and Vincent's batteries [24, 29]. Carlos is a synthesizer. Then off goes Miranda to talk with Barry; she must be an enthnographer, a Turkle "wannabe." Soon, Miranda will be overloaded, for she has initiated topics with Natalie, Jacob, and even Barry. Yet, while Miranda wonders about the strange words in the spelling checker utility, Peter is deep in Turkle's conclusions and composing a thorough response to the first topic, the first purpose.

Ariel [34:4:28]:
I agree with Natalie... I don't know much about child psychology, but I thought, from reading the book, that

computer games do not facilitate earlier development of psychological questions, I think of children as always asking these kinds of questions, since they start to talk.

Natalie [35:4:34]:
Ariel, thanks for coming to my rescue!

Carlos [36:2:19]:
Ariel, I think you are correct, like Tenley mentioned, children are very imaginative and like to possess a sense of control. Recall the little girl who had no friends, then became friends with the speak and spell. . .like the other kids who would bully her around she would bully the machine by demanding it to say something. The same with the child who removed the batteries to "kill" the "bug" in the "say it" program. I think I have observed the same behavior with children who have no siblings but do have a dog. . . they tend to force orders on their pets like "sit" "out" etc. since there is no one else to "boss" around. Yes, you are correct.

Miranda [37:7:33]:
Barry -- I liked your message, even though I was eavesdropping. I haven't seen or heard the word jehozaphat in while. Is it in spelling checker?

Peter [38:2:1]:
At the end of the chapter, Turkle asserts, "Thought and feeling are inseparable." Oh really? As we saw from Costanzo, computers are capable of (limited) thought, yet no one would deny that these machines cannot really feel anything. Any "feeling" that could enter into the computer's thought processes would have to be a digitized, programmed mortality (hence, not its own.) Turkle argues that if thought and feeling are separated, (a) they become "mutually exclusive," and (b) "the child's sharpened distinction between intellect and emotion can easily lead to a shallow and sentimental way of thinking about "feelings." Give children some more credit, please. Yes, computers can (and do) separate thought from feeling, but children are not hapless victims of a "garbage in, garbage out" relationship with a thought-only entity. Rather, children (and adults!) have much to learn from a creation that indeed can separate the two.

Jacob responds to Miranda's concerns about the Third world. Barry responds to Miranda's spelling checker query and drops in his basic philosophy of modern life: we are what we watch. Natalie has been thinking about Jacob's situation in South Africa. Miranda decides to offer as "proof" to Natalie and Ariel her own experiences with her sister with "conscious" dolls doomed to living out their lives in dusty corners of attics. Not bad pathos, Miranda. Meanwhile, Vincent, Peter, Miranda, Tenley, and I continue debating thought and feeling matters. I am overcome with a graphic (if not a communicative) urge.

Jacob [39:3:31]:
Miranda, thanks. Yes, I'm sure there are such powers around, who would seek to protect and distribute the resources of power in self-interested ways. But from the point of view of third world students, there is a barrier of suspicions, i.e. it's not really the case that they are scrambling to get hold of these resources.

Barry [40:2:37]:
Miranda--"I doan thin so," in the words of Ricky Ricardo. As you may guess--I am what I watch. I got "jehozaphat" from Yosemite Sam.

Natalie [41:5:23]:
Jacob, I find it interesting what you said about gaining in humanity by being part of a communicative network. However, I wonder if I could feel like that. I sometimes felt like we lost some humanity here by talking to the screen instead of to each other.

Miranda [42:8:28/34/36]:
I think that children do think a lot about what is alive and what the implications might be. My sister was so convinced that her dolls had consciousness that she lined them up on her pillow and slept on the floor. Little girls love to read biographies of dolls who have lives of adventure followed by years thrown in dusty corners of an attic.

Vincent [43:5:38/1]:
Peter--there are those who might disagree with you. Some would consider thought as an exclusively human procedure, linked to all other human traits, including emotion. But I don't think that leads to Turkle's conclusion. Especially children, I think, consider feelings as part of experience and not clouded with these abstract considerations.

Miranda [44:9:21/38/1]:

```
I think Turkle makes a strong case that children are
thinking in new ways about psychology -- for example
the question of whether cheating can really be called
cheating if there is not intention and self-
consciousness. Where could they previously have
experienced anything like cheating from a non-living
entity? It's all new! Too bad Piaget is not around
when we really need him. . .

Tenley [45:6:38]:
All of this about computers being alive or not makes
me think of people who think of computers as not
creative. I think kids like computers so much because
they are mainlined into creativity and thus can
suspend the adult misgivings of thinking/feeling etc.
. . and just enjoy their "imaginary" friends:
electronic or imagined.

Hugh [46:6:38]:
Peter, the thought and language business. Boy, every
time I begin to think of thought and feeling as
separate entities, a little Eastern imp whispers in my
ear: it's not that simple. I wrote in my journal
recently about the yin and yang of thought versus the
yin and yang of feelings--as a way to break through to
"staying on the same intellectual continuum":

    Yin----------THOUGHT----------Yang
    Yin----------FEELING-----------Yang, and
    Yin----------LIVING-------------Yang, etc.

Does this make sense to anyone? If so, let me know.
I am in search of a multidimensional model of
cognition and whateverfeelition.
```

With this yin/yang model introduced, the Interchange took off in other directions, including one giant rhyming diversion. Watch what happens to Ariel's "baboons" once Carlos misreads Ariel's response. Like sharks to blood in the ever salty sea, they came. . . .

```
Miranda [47:10:41]:
I guess it's a real boon to the computer industry that
parents don't have time for their children. The
computer interacts with children, providing much more
"touch" than TV; it is a better baby-sitter than most
live ones, except that it doesn't require the child to
be particularly sociable, only nonviolent!

Vincent [48:6:47]:
```

Miranda- I guess then we need not fear a lack of techno-junkies to maintain our system of technologies.

Marilynn [49:3:46]:
Sorry Dr. Burns, I left my model at home today.

Carlos [50:3:45/44]:
Tenley, I agree with you, that is precisely what Turkle points out in how they perceive a computer, they offer it human qualities:
INTELLIGENCE, when they see that it talks, thinks, and remembers; FEELINGS, when it makes them happy, sad, etc. sounds to praise the child or otherwise, and finally the MORALITY issue, "the computer cheats." Like Miranda said, too bad Piaget is not here to see all this!

Barry [51:3:46]:
Hugh--does your taoist model imply that writing collapses that distinctions between yin (creativity) and yang (analysis)-- I may have the alignment of yin and yang reversed--? Or is it that the dynamic tension of the opposing forces propel writing?

Barry [52:4:51]:
Hugh-- er . . . propel*s*

Ariel [53:5:50]:
Carlos, it seems to me that everyone attributes almost everything with human attributes, not just computers. I recently saw a show about baboons with Loren Green and he "humanized" EVERYTHING they did.

Miranda [54:11:46]:
Hugh, I think you're on to something with your chart of yins and yangs. Yes, the distinctions may be illusions, products I suspect of our Newtonian world view which from what I understand is obsolete. The distinction between thought and feeling doesn't make much sense either in light of our current understanding of brain chemistry. One distinction that still looms large to me is that of something that can have values and something that can't. I don't know where this fits on the yin yang scale, but probably right in there with LIVING. I am taking a course in cognitive psychology, and apparently Piaget is "out." I wonder how a state of the art cognitive psychologist, therefore, would view all this. Personally, I still like Piaget, but I'm beginning to see some of the ways his theories don't always hold up.

Carlos [55:4:53]:

Ariel.....BALLOONS? I cannot imagine, but maybe I will try and do the same tonight and see how many attributes I can find.

Natalie [56:6:?]:
I once read a book by Hoimar von Ditfurth, a philosopher-biologist, who said that feelings are only reflections of a purely biological process in our bodies. What do y'all make of that?

Hugh [57:7:51/52]:
Barry, yes, writing collapses such distinctions, or at least compresses them to me. Discovery, heuristics, yin! Recovery, hermeneutic, yang! Now let's add time cycles, very fast time cycles, and recurse! That is what the faster and faster and faster computers do: imitate that recursion process such that we "believe" in their "protoquasimeta" life force. Why do I think like this about a machine? We are talking about carburetors and toasters and microwaves, aren't we? Who would ever want to talk about yin and yang of making toast, anyway? [Making "taoist"?] So what's the big deal with making prose and writing and understanding and using an electronic device. Jumpin' GeorgeBushosavat!

Ariel [58:6:55/53]:
Carlos! BABOONS, I said.

Vincent [59:7:46/43/38]:
Cool.... computers will revive western thought after all.

Hugh [60:8:58/55/53]:
SALOONS?

Miranda [61:12:56]:
Natalie--I suspect that Ditfurth is correct, except that I would remove the word "only." The biological processes in our bodies are pretty complex and probably related to more in the world than we realize.

Ariel [62:5:60/58/55/53]:
BAFOONS

VINCENT [63:8:59/46/43/38]:
Let me get this straight... are we being dehumanized or are computers being humanized? Or do we care about humanization anymore? I'm confused as to what we are saying...

Barry [64:5:62/60/58/55/53]:
Bassoons

Miranda [65:13:64]:
Bassoons?

Carlos [66:5:58]:
Ariel...sorry, no wonder that threw me off, but I'll still try the balloons!

Marilynn [67:4:50]:
Miranda--I can see where all of this would integrate into Piaget's qualitative development chart. Five and six year olds clearly at the preoperation stage, by definition, would describe computers the way that Turkle has depicted them. (Based on my severely limited knowledge of Piagetian theories.) It seems that as the abilities of conservation and abstract reasoning are acquired, then they will comprehend the abstract qualities of computers.

Vincent [68:9:59]:
Computers are only abstract if you know the way they work.

Ariel [69:6:66/64/62/60/58/55/53]:
Well, Hugh, is this over?

Yes, Ariel, the session is over. But the negotiated *topoi* covered purposes, acts, scenes, agents, and agencies. Moreover, the networked epiphanies were integral, harmonious, and radiant. Jumpin' jehozaphat! This session lasted about forty-five minutes. Eleven people participated. Sixty-nine messages were entered.

Again, in discovering collaborative standards and negotiating within a realtime community, the participants constructed an interpretation, reached for some consensus and, perhaps, even enjoyed the experience.

Implications and Research Issues

To review, some "act," "agency," and "agent" predictions are easier to make:

(1) Technology assessment methods and tools will rapidly evolve.

(2) Computers will play greater and greater roles in day-to-day assessment activities.

(3) Computers will become commonplace communication tools--tools widely used by information consumers and knowledge producers.

(4) Assessment of the impact of advanced computer technology impact on contemporary communication, especially ways peopl- interact, will require study.

(5) Understanding how knowledge is constructed or deconstructed, used or abused, shared or not shared will be critical.

(6) Emerging technologies will provide self-referential tools for validating theoretical performance and improving practices.

(7) Electronic interactivity will open new ways for investigating how people discover, recover, organize, pattern, edit, and deliver information.

Other "purpose" and "scene" predictions are often harder to support:

(1) Technology will make real and significant differences.

(2) Technology-based education will revolutionize classrooms and change forevermore how people communicate.

These bolder predictions, however, are easily the more fascinating. Experts may continue to disagree about the extent to which information technology will be useful in contemporary research, but coming to terms with innovation, with purpose, with situations, with uncertainty, with consequences, with technological "dispositions" in a complex world, with respect for another person's ideas--such "topoi" have long been the challenge and the opportunity for a contemporary people no matter what century they lived in.

Networked software provides many design options for personal, highly collaborative interactions, filled with moments of becoming more of an expert, of understanding what was not understood, of being challenged to know and articulate important things in personally useful and publicly useable ways. The future of collaborative, social technology assessment instruments will allow realtime evaluation, immediate access to information, new understandings about the dynamics of knowledge, and perhaps a simultaneous software engineering design-on-the-fly toolkits.

Recovering the "topoi" and discovering the integrity, harmony, and radiance of a technology will represent a major part of the work as we construct an estimate of the future of technology assessment. *Integritas, consonantia, claritas*--the field of technology assessment needs such "new" insights.

In Alice's excellent adventure, I'll never forget how the Mock Turtle summarized his basic education in "Reeling" and "Writhing." Thus, the method behind this paper's madness--its reeling and writhing--has been to reacquaint us with the power of language and thought. Isn't it wonderful how we humans recover our ideas, and how language forms meanings? Isn't it novel that we could use technology to negotiate meanings in community in order to discover that we actually mean what we say. Saying, as the Mock Turtle said, "I mean what I say" is possible. But a finer way to discover is not to assert an assessment but to negotiate the purposes, acts, scenes, agents, and agencies of technology assessment. Instead of competing against one another, users find strength in collaboration, helping one another. That's the challenge of collaborative learning generally and even more so with these emerging technologies needing to be assessed and needing to be used wisely and well. Whatever else happens to technology assessment, one thing is certain: a basic education in technology assessment should not depend more on reeling and writhing than on reading and writing in collaboration. Negotiate the topics. Network the epiphanies.

References

Aristotle. (1954). *The Rhetoric and the Poetics of Aristotle.* W. R. Roberts and I. Bywater, translators. New York: Modern Library.

Barrett, E.(Ed.) (1989). *The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information.* Cambridge, MA: The MIT Press.

Barrett, E.(Ed.) (1988). *Text, Context, and Hypertext: Writing with and for the Computer*. Cambridge, MA: The MIT Press.

Bump, J. (1990). "Radical Changes in Class Discussion Using Networked Computers. *Computers in the Humanities* 24: 49-65.

Burke, K. (1969). *A Grammar of Motives*. Berkeley, CA: University of California Press, 1969.

Burns, H., and Culp, G. (1980). "Stimulating Invention in English Composition through Computer-assisted Instruction." *Educational Technology*, August, 5-10.

Carroll, Lewis. (1960) *Alice's Adventures in Wonderland & Through the Looking-Glass*. New York: New American Library.

Costanzo, W. V. (1989). *The Electronic Text: Learning to Read and Reason with Computers*. Englewood Cliffs, NJ: Educational Technology.

D'Angelo, F. J.(1975) *A Conceptual Theory of Rhetoric*. Cambridge, Massachusetts: Winthrop.

Joyce, J. (1968). *A Portrait of the Artist as a Young Man: Text, Criticism, and Notes*. Chester G. Anderson, (Ed.) New York: Penguin Books.

Kinneavy, J.L. (1971). *A Theory of Discourse: The Aims of Discourse*. Englewood Cliffs, NJ: Prentice Hall.

Winograd, T. & Flores, F.(1986) *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, NJ: Ablex.

Turkle, S. (1984). *The Second Self: Computers and the Human Spirit*. New York: Simon and Schuster.

Vallee, Jacques, Johnansen, Robert and Spangler, Kathleen. "The Computer Conference: An Altered State of Communication?" In Stewart Ferguson and Sherry Devereaux Ferguson, eds., *Intercom: Readings in Organizational Communication*. Rochelle Park, New Jersey: Hayden Book Company, Inc., 1980: 290-300.

Zuboff, Shoshana (1988). *In the Age of the Smart Machine: The Future of Work and Power*. New York: Basic Books.

Assessing Technology in Assessment

Henry Braun
Educational Testing Service

Introduction

A powerful tool such as a personal computer (or some other product of modern technology) often seems to take on a life of its own, overshadowing the task to which it is applied. For this reason alone, it is useful to step back occasionally and reflect both on the actual contributions of the technology and on the costs associated with its use. In so doing we may hope to learn better how to harness technology for our purposes.

This chapter considers the role of computer-based technology in assessment particularly for the licensing and certification of professionals such as architects, physicians, and engineers. It should be noted at the outset that whatever the contribution of this chapter, it rests on informed speculation rather than on a completed formal evaluation. The case study that underlies the discussion concerns the development of computer-based

Acknowledgments: I would like to thank Randy Bennett, Richard Devore and William Ward for their helpful comments on an earlier draft of this paper. I also want to take this opportunity to express my appreciation to my co-workers at ETS, members of the staff of the National Council of Architectural Registration Boards (NCARB) as well as the practicing architects who have volunteered their time to the Architectural Simulations Project, for providing such a stimulating, collegial environment within which to carry out the work that is briefly described herein. The work reported here has been jointly supported by NCARB and ETS.

Introduction (cont.)

simulations of architectural practice. At the present time, three years of work have been completed and several more are planned. Despite the somewhat fragmentary nature of the results, there has been some opportunity to consider the impact of technology. Hopefully, our thoughts and reflections will prove useful to others.

Technology in the Assessment of Professionals

The purpose of assessment in licensing and certification is to make accurate and reliable decisions as to whether a candidate has met certain standards of competent performance, ordinarily involving a range of higher order cognitive skills as well as the mastery of an extensive knowledge base. In evaluating the role of technology in this process, we must first ask several key questions and then examine the way in which technology affects our responses. The key questions are: What aspects of performance can (or should) be tested? Is an assessment program feasible? Are the inferences to be made valid?

Perhaps the first contribution of the new technology has been to stimulate many professions to reexamine their assessment process. The advent of relatively inexpensive personal computers connected through local area networks, as well as peripherals such as videodisc players, appears to promise both enhanced efficiency and greater comprehensiveness. Concurrently, segments of the testing industry see an opportunity to introduce new kinds of probes that can be organized in novel ways, making it feasible to validly assess a broader range of competencies. Moreover, technology may well facilitate quicker reporting and the provision of useful feedback to the candidate.

However, it is also recognized that there are potential problems with the application of technology to assessment. These problems include cost, delivery, program maintenance and equity. Each point will be examined in turn. At the same time, it is important to take an ecological perspective. It may be that the introduction of technology can have systemic effects, both positive and negative, that are not captured by a more reductive analysis. In a later section, we will discuss a particular benefit that appears to result from one technological innovation.

The Architectural Registration Examination

A case study in the use of technology in assessment involves work underway at Educational Testing Service (ETS) on the use of computer-based simulations of architectural practice. The role of the simulations is to carry out performance assessment of aspiring architects through creating on the computer a version of a natural work setting. By using technology, it is possible to create a dynamic environment in which candidates can work much as they would in a real office. Moreover, advances in artificial intelligence may allow automated scoring of complex constructed responses.

Inasmuch as testing and computer technology are relatively new bedfellows, it seems safer to reason from the specific to the general, rather than the other way around. Accordingly, let's begin with a brief discussion of the context in which the Architect Registration Examination (A.R.E.) operates.

Architects, like physicians, cannot simply hang up a shingle once they have completed their academic training. In most states, they must enter into an internship program and, after three-to-four years, are eligible to take a series of examinations, the A.R.E., sponsored by the National Council of Architectural Registration Boards (NCARB). The purpose of the examination is to protect the public's health, safety and welfare. Consequently, the component tests focus on essential elements of competent practice, rather than on areas like aesthetics of design. Nonetheless, candidates have some opportunity to display creativity by accomplishing tasks requiring problem-solving under constraints.

At the moment, there are nine examinations, seven employing paper-and-pencil based multiple-choice testing and two employing graphical-response testing, involving the exhibition of design skills. Student solutions in the latter two examinations are scored by trained pairs of architects randomly drawn from juries specially convened for this task.

More than three years ago, NCARB asked ETS to begin conversion of the multiple-choice examinations to computer delivery. That process is nearly complete and some small-scale operational testing has been carried out. Moreover, a new form of test administration, called computerized mastery testing (CMT), has been developed (Lewis and Sheehan, 1989). With CMT, candidates are required to take two "testlets" (Wainer & Kiely, 1987), each containing ten-to-twenty-five items, depending on the particular test. Each testlet reflects the test content specifications. After completing the two testlets, a determination is made as to whether the candidate has passed, failed, or whether there is insufficient information to determine the outcome. In the latter case, additional testlets are administered until sufficient information to make a decision has been obtained, or a predetermined maximum number of testlets has been administered.

In addition to conversion of the original test battery, NCARB also asked ETS to develop a new computer-based simulation of architectural practice. The new test would be designed to complement the existing battery and to assess, in a realistic setting, the higher-order skills considered essential to the competent practice of architecture. These include analytic and creative, as well as management, coordination and communication skills. As presently conceived, the test will consist of three or four simulations, each based on a different architectural project. A simulation, in turn, will be composed of three to four vignettes drawn from a project script. Each vignette presents the candidate with one or more related tasks and the candidate must produce a solution that is relatively unconstrained and entirely uncued.

The immediate goal of the technologist is to create a computer-based version of the architect's work setting, including the resources and design tools typically available in that setting. The environment should permit the candidate to develop and modify solutions in a relatively natural fashion.

The system we have developed employs two monitors. A high-resolution monitor is used as a model office. It contains icons reflecting three types of resources: a "bookshelf," a "drafting table," and a "file cabinet." The bookshelf contains excerpts from standard architectural references and codes. The drafting table contains blueprints and other drawings that are specific to the particular project at hand and, similarly, the filing cabinet contains written materials that are specific to the project. Using a mouse, the candidate may access any of the resources at any time, may page through either reference volumes or project materials, may record copies of those materials in an on-screen notebook and access the notebook directly at any time during the simulation.

The other monitor, called the work screen, represents the candidate's work space. Here problems are presented, and the architect carries out his or her design activities in response to the tasks posed in the vignette. For each vignette, a set of icons are available for the candidate's use. Using the mouse to activate the different icons, the candidate carries out the different functions necessary to producing a solution. A few icons, such as "HELP" or "START OVER," are common to all vignettes. Together, the tools available to the candidate constitute a simple system for computer-aided design. It is worth mentioning that, using the mouse, the candidate can move smoothly from one monitor to the other; hence it is very easy for the candidate to employ the resources in the model office at any time during the course of solving the problem at hand.

In one vignette, the candidate is asked to design a block diagram (a kind of preliminary floor plan) for a particular building, say a library. He/she is provided with a set of design elements which must be arranged on a particular site in a way that is responsive to the demands of the program. The spaces comprising the building are represented by squares whose areas are proportionately scaled to the areas described in the project script. The introductory screen is illustrated in Figure 1. The design elements are initially placed in a horizontal strip along the top edge. Along the left edge are a series of icons which are activated when clicked on with the mouse. Each icon allows the candidate to perform a certain function.

For example, one icon allows the candidate to move ("drag") the design elements onto the site, which covers the central portion of the screen. A second permits the height and width of the element to be adjusted while keeping the area constant. A third causes the design elements to be rotated on the site in order to achieve certain requirements (i.e., to create views or to avoid knocking down nearby trees). Another allows the candidate to indicate how the circulation in the building will occur, either between spaces or along public corridors. Obviously, the number of potential solutions, even for a relatively small number of design elements, is effectively infinite, making scoring a nontrivial task. A typical solution is presented in Figure 2.

Another vignette requires production of a structural schematic drawing. In this vignette, the candidate must indicate how he or she would lay out the structural frame for a building or a particular section of a building. The introductory screen for this vignette is presented in Figure 3. In this case, there are no preset design elements. Instead, the candidate must call on different icons depicting load-bearing walls, columns, beams and joists, in such a way as to provide a viable and economical structural frame. A typical solution is displayed in Figure 4. Again, even for a relatively small problem, the number of potential solutions is very large.

Assessment of the role of technology requires that we return to the questions mentioned at the beginning of the chapter: Does the use of technology extend the range of competencies that can be assessed and does it enhance the validity of the decisions that are made on the basis of the examination? Furthermore, does it do so in a feasible manner -- technically, economically and psychometrically? A meaningful discussion requires a clear view of the goals of the examination in the context of the real-world setting in which the examination operates.

Validity and Feasibility

The concept of validity has undergone substantial evolution over the last forty years (Angoff, 1988). The modern conception (Messick, 1989) declares that validity concerns the appropriateness of the inferences and decisions that are made on the basis of the test. In the present setting, this suggests the following questions: Are we measuring the appropriate constructs of architectural practice with the use of this test? How are these constructs related to those previously measured? Are the new constructs measured reliably (with sufficient replication) and accurately (with effective scoring programs)? Are the decisions made based on the augmented battery of tests superior to those based on the original battery? Finally, do these decisions create or exacerbate inequities among subgroups in the candidate population?

Feasibility is concerned with whether the examination can be developed, delivered and maintained at a reasonable cost and whether the appropriate psychometric procedures can be developed and applied to support the program. It is important to note that both feasibility and validity must be considered from conception through implementation and beyond, and that empirical evidence on feasibility usually precedes validity evidence.

Let us first consider some of the issues associated with feasibility. One of the ostensible intermediate goals of any simulation is to achieve fidelity to real life (Fitzpatrick & Morrison, 1971). In an examination of architectural skills, realism -- in both the resources available to the candidate and the nature of the exercises presented -- is desirable. Nonetheless, true fidelity can usually be obtained only at great cost, where cost must be thought of in a general sense.

For example, providing an extensive set of design tools will increase development costs, as will constructing complex vignettes involving many design elements. There are hidden costs as well. Increased complexity necessitates the need for more elaborate tutorials and longer practice time to prepare candidates to take the examination. One solution lies in identifying those features of the real-life setting that are essential to the assessment of the desired constructs. The simulation should then be designed to be as simple as possible while capturing those critical features. (This may well involve empirical research.) Thus, practical assessment through simulations will depend as much on a talent for knowing what to leave out as what to put in.

In the introduction of any new technology such as computer-based simulations, there are large initial costs that must be borne by the client and the developer. These costs often represent a substantial barrier to the development of new kinds of tests and must be taken into account when considering the viability of the technology. Computer-based tests also necessitate extended administrations due to the lack of a sufficient number of computer terminals with which to test the candidate population at one time. These extended administrations entail substantial costs for equipment leasing, multiple exam locations, staffing and test security measures.

While candidates benefit from more flexible testing schedules, security is more complicated than simply hiring a number of proctors to make sure that examinees don't cheat. The security issue revolves around the fact that if the examination is given repeatedly over a period of time, those taking it later will have the advantage of talking to friends who took it earlier. Presumably, the exam will be easier for the candidates tested later in the administration period. One way of countering this scenario is to create a large library of simulations so that it is very unlikely that two people will take exactly the same set of simulations. Unfortunately, the construction and maintenance of this large library can be extremely costly and, as a result, if the examination is to be feasible, efficient test development procedures must be established.

Another technical issue concerns the generation and maintenance of scoring programs. As mentioned earlier, for vignettes that result in complex constructed responses, the number of potential solutions is usually effectively infinite so that it is not possible to score a response by simply comparing it to a template of the perfect solution. In fact, there may be many classes of ideal solutions (or, for that matter, many classes of solutions meriting partial credit), each class having an infinite number of members. Consequently, an acceptable scoring program must behave like a mini-expert system. That is, it must be able to decompose and reason about the candidate's solution and then place each solution at the appropriate point on a score scale.

These desiderata pose several challenges. First, building an expert system requires extensive knowledge engineering as well as substantial empirical testing to assure accuracy and comprehensiveness. Both activities are labor intensive and time consuming. Because of the large library of vignettes that must be maintained, these systems have to be easily modifiable to handle related problems, as the effort needed to construct a new system for each vignette would be enormous.

Interestingly, building algorithms for assigning partial credit scores to solutions also involves considerable effort in setting and validating standards. While there has been some work in the psychometric literature (Andrich, 1978; Wilson, 1990) on developing relevant models, there has been comparatively little reported on systematically eliciting experts' judgments for such purposes (though some organizations, like ETS, have considerable practical experience in this area). Accordingly, there are a number of issues to be resolved. Among them are: (i) Determining the optimal number of score categories; (ii) Generating usable generic descriptions of each category to facilitate the establishment of specific scoring guidelines; and, (iii) Developing methods for monitoring the alignment of scoring standards among different vignettes. Each problem will require both analytic and empirical contributions for their resolution.

The introduction of tasks requiring complex constructed responses also raises many interesting questions for the measurement community. Because candidates need more time to respond, it is essential that the maximum amount of information be extracted from the solution. Yet logic suggests that the task of developing appropriate psychometric models will be simplified if the number of features retained is kept as small as possible. Resolving the tension between these two demands will be critical.

It is also generally accepted that current psychometric theory does not adequately address the needs of simulation tests. Most likely, we will have to rethink classical notions like reliability, rather than engage in simple modifications of existing measures. Moreover, a new framework for making pass-fail decisions must be developed. Having completed the exam, the candidate presents a profile of vignette scores. A defensible decision-theoretic framework for mapping profiles into the pass or fail categories is essential. Developing such a framework will require close collaboration between psychometricians and expert architects.

The difficulty is due to the fact that different vignettes may well test different constellations of skills, mastery of which is considered essential to competent practice. If the public's health, safety and welfare is to be protected, a strong performance in one area should not be allowed to compensate for a poor performance in another area. Thus, adding the vignette scores and locating a convenient cut point on the resulting scale would not be a desirable procedure. More refined methods are needed.

Establishing the validity of a licensing examination is an arduous and on-going task. New conceptions of validity (Messick, 1989) emphasize the importance of construct validity as the overarching conception in the validation argument. Demonstrating high correlations between test scores and reliable measures of job performance would constitute evidence for validity, but is for the most part impractical. In fact, practitioners (Kane, 1982) have argued that it is nearly impossible to develop suitable criteria with which to conduct classical predictive validity studies.

The alternative is to imagine the test as a vehicle for assessing the skills and abilities essential for competent practice. The actual questions on the test represent a sample of the contexts in which the candidate might reasonably be expected to exercise those skills and abilities. Both Messick and Kane (and other authors, as well) provide guides to defensible validation strategies.

In developing the NCARB exam, we have not yet begun to deal formally with validity issues. Our premise has been that proper construction of the test provides a sound foundation for establishing its validity for appropriate uses. For computer-based tests, proper construction includes attention to such matters as building a user-friendly interface and informative tutorials as well as more traditional concerns related to the nature and level of skills tested.

At the moment, we are working with a number of different task analyses that have been commissioned by NCARB, as well as with committees of expert architects to ensure that the exercises and activities embedded in the vignettes are relevant and that they are, on the face, testing the kinds of skills -- for example, analytic and creative skills -- that are appropriate for this type of exam. Once a larger number of vignettes have been constructed and combined to form simulations, we will begin to collect pilot data. On the basis of these data, we will initiate both cognitive and psychometric research to gather evidence concerning the constructs measured and how they relate to those measured by the current multiple-choice exams. Of course, these data will also enable us to address such issues as timing, accuracy of scoring and fairness.

A serious concern, especially with computer-based tests, is that performance should not depend on a construct-irrelevant factors. For example, prior familiarity with computers or with computer-aided design ought not confer an advantage in test performance. Consequently, the simulation should employ a simple, generic design system that is easy to master after a relatively short trial period. Furthermore, any differences in performance by gender, race or ethnic group must be carefully scrutinized to insure that they do not arise because of factors unrelated to the test's purpose.

Ecological Approach to Assessing Technology

It is important that any attempt to assess the contribution of technology to the practice of testing take account of potential system effects that might not be captured by a series of focused questions formulated a priori. An analogous situation arises in the classroom, where the introduction of computers may change the nature of the teacher-student dynamic as well as specific student-centered learning behaviors.

The systemic validity argument (Frederiksen & Collins, 1989) declares that good tests will force the school system to adapt in educationally sound ways in order to meet the new goals. There may well be such an effect on schools of architecture when the simulation test becomes operational, although the gap between the end of schooling and the taking of the test is rather substantial. I would like to argue that it is more certain that the new technology will have profound effects on the system that produces the test!

As we gain more experience with this kind of test, it becomes clearer that a new partnership must be forged among test developers, cognitive scientists, computer scientists and psychometricians. Successful development of performance assessments (sometimes referred to as "authentic assessments"), will require close collaboration of these different specialties in order to ensure that a viable examination emerges. For example, rather small differences in the presentation of a vignette can have substantial implications for the complexity of scoring. Consequently, an efficient iterative process must be worked out so that the finished product reflects the best attainable compromise between, say, fidelity and feasibility.

Perhaps the most exciting (unanticipated) benefits arising from the introduction of technology lie in the effects of the development of automated scoring through mini-expert systems. By virtue of having to create computer programs to score complex constructed responses, test developers are forced to rationalize and stabilize a set of test specifications which in the past may have been vague and ambiguous. (This should contribute to construct validity as well.) A clear definition of the vocabulary and universe examined by the test will be essential in building an operational scoring system. Once the test specifications are developed in a way that can support automatic scoring, there exists the basis for creating a system which provides greater stability of test vignettes from one administration to the other.

Equally important is the fact that by developing such systems, approximately 90% of the work required for diagnostic feedback will have been done. That is, once automated scoring is on line, useful diagnostic feedback can be sent to the candidate with little extra effort. This innovation can then be transferred to both the school and workplace.

Conclusions

The preceding analysis has suggested some of the ways in which the introduction of technology can aid assessment. Nonetheless, to realize this potential will require a great deal of hard work on the part of researchers in many disciplines, as well as some unavoidable trial-and-error. Although the discussion has focused on the example of architecture, most of the points raised should be relevant to other professions as well. Naturally, though, each profession will require certain specialized features in the assessment program.

In medicine, for example, time pressure is often a critical element in medical decision making. A proper simulation should recreate the temporal character of medical practice. In fact, the simulations now under development by the National Board of Medical Examiners incorporate this aspect of practice.

More important, it is essential that simulations for assessment be clearly distinguished from simulations for training. When training is the major purpose, the balance between fidelity on the one hand and cost or feasibility on the other will shift toward the former. Nonetheless, it is already clear that simulations incorporating computer technology provide an exciting and practical means of assessment. The testing profession must hope that by developing assessment instruments embodying the precepts of systemic validity, it can begin to change the attitudes of the public, as well as segments of the educational community, towards the concept of standardized testing as a fair way of generating information for decisions.

REFERENCES

Andrich, D. (1978). A Rating Formulation for Ordered Response Categories. Psychometrika, 43, 561-573.

Angoff, W. (1988). Validity: An Evolving Concept. In H. Wainer and H.I. Braun (Eds.), Test Validity. Hillsdale, NJ: Lawrence Erlbaum, 19-30.

Fitzpatrick, R., & Morrison, E. (1971). Performance and Product Evaluation. In R.L. Thorndike (Ed.), Educational Measurement, 2nd Ed. Washington, DC: American Council on Education, 237-270.

Frederiksen, J., & Collins, A. (1989). A Systems Approach to Educational Testing. Educational Researcher, 18, 27-32.

Kane, M. (1982). The Validity of Licensure Examinations. American Psychologist, 37, 911-918.

Lewis, C., & Sheehan, K. (1989). Using Bayesian Decision Theory to Design a Computerized Mastery Test, (in press).

Messick, S. (1989). Validity. In R.L. Linn (Ed.), Educational Measurement, 3rd ed. New York: ACE and Macmillan, 13-103.

Wainer, H., & Kiely, G. (1987). Item Clusters and Computerized Adaptive Testing: A Case for Testlets. Journal of Educational Measurement, 24, 185-201.

Wilson, M. (1990). An Extension of the Partial Credit Model to Incorporate Diagnostic Information. Berkeley: University of California Press.

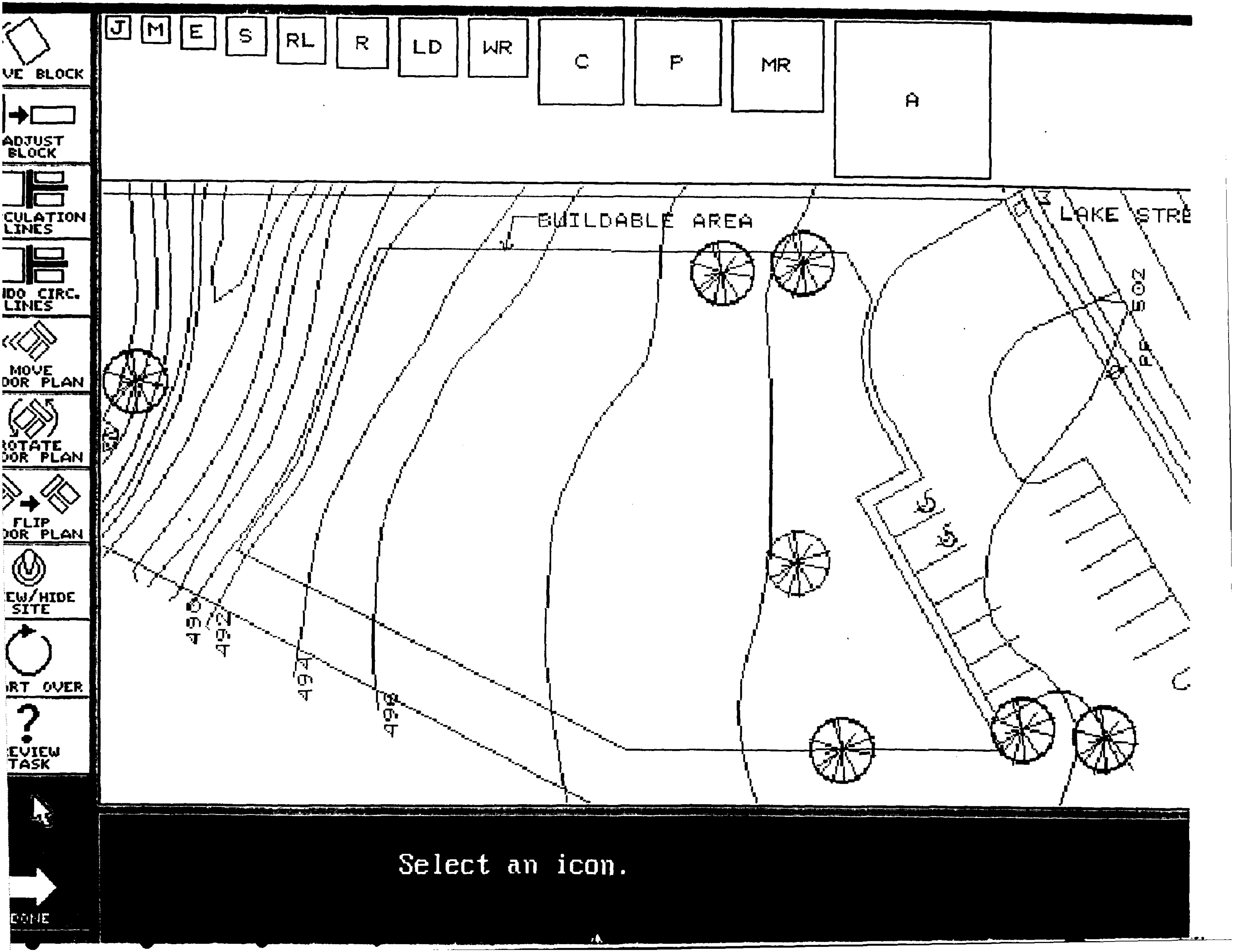
CAPTIONS FOR FIGURES

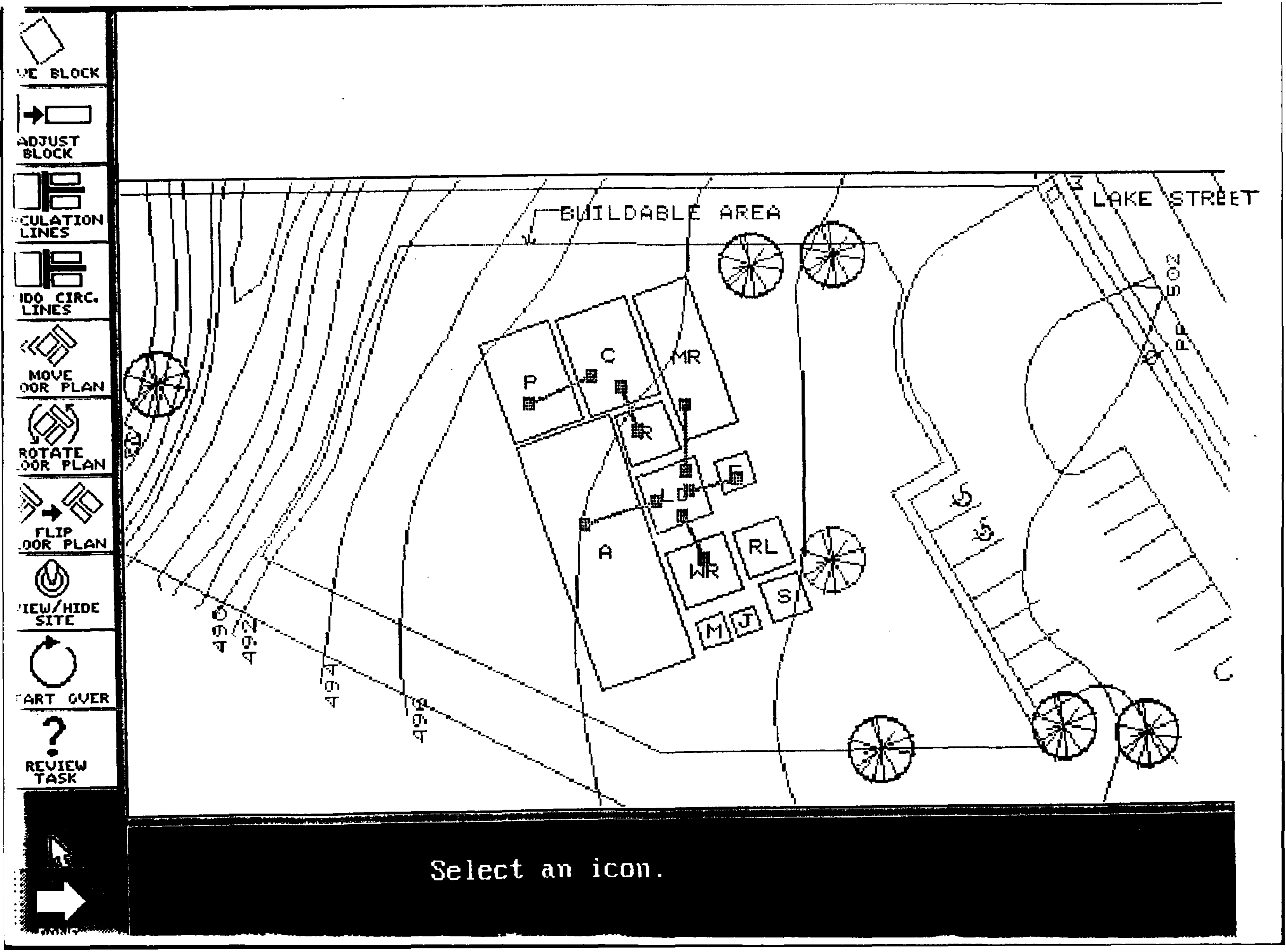
Figure 1: Introductory screen for vignette for designing a block diagram.

Figure 2: Screen displaying typical solution for block diagram vignette.

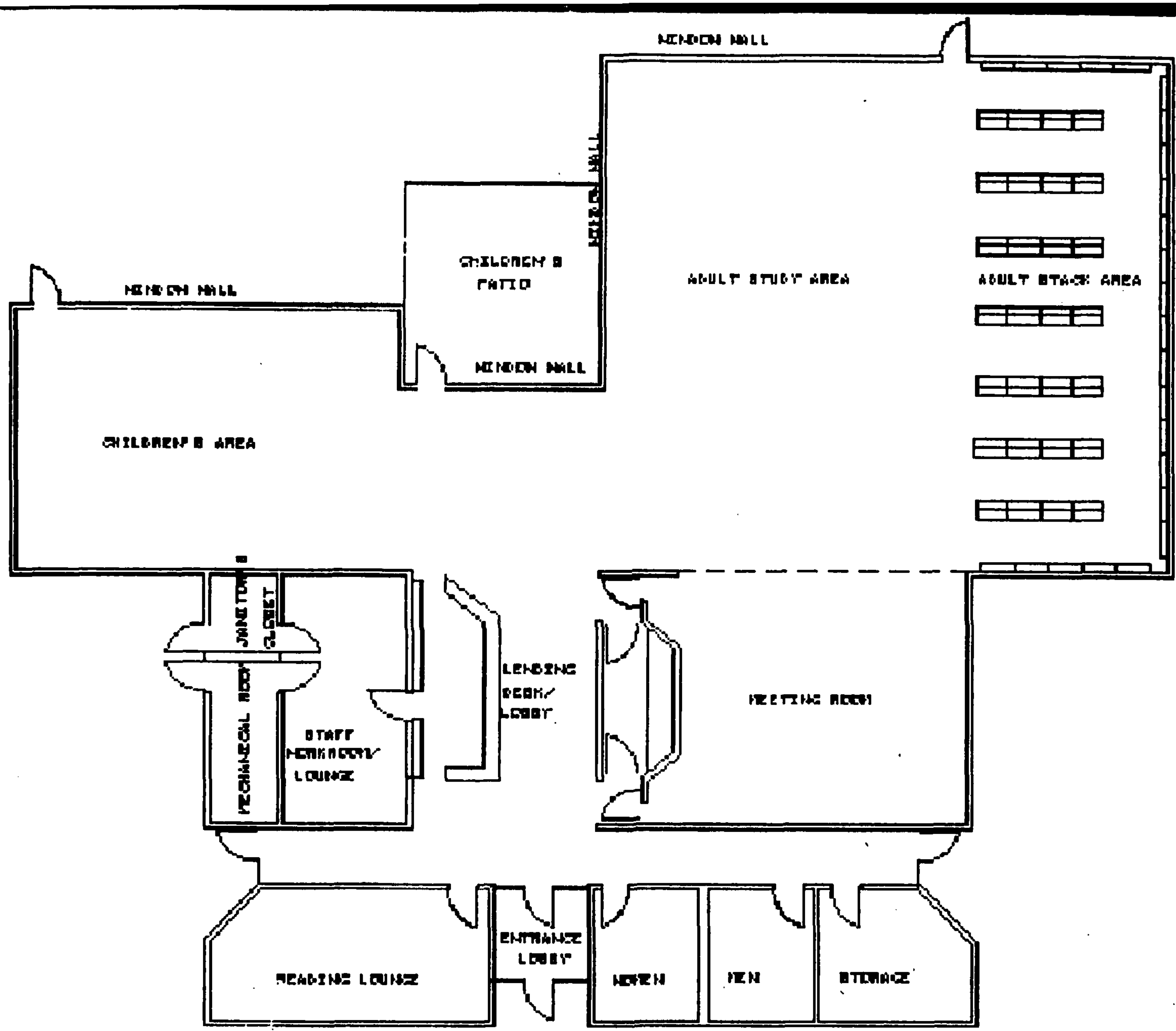
Figure 3: Introductory screen for vignette for designing a structural schematic diagram.

Figure 4: Screen displaying typical solution for structural schematic diagram vignette.

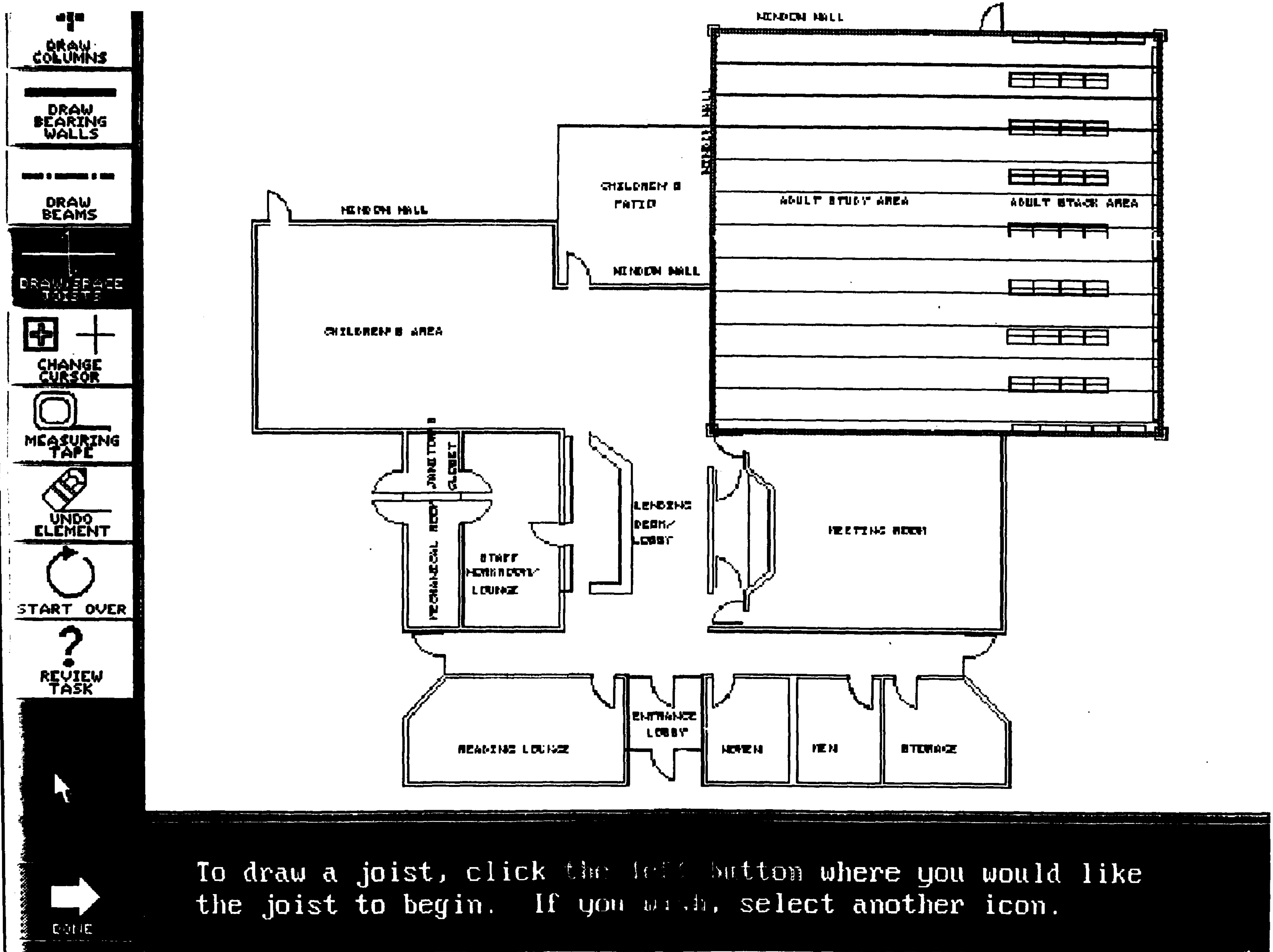

VE BLOCK
ADJUST BLOCK
CULATION LINES
IDO CIRC. LINES
MOVE
DOR PLAN
ROTATE
DOR PLAN
FLIP
DOR PLAN
EW/HIDE SITE
RT OVER
EVIEW TASK
DONE
J
M
E
S
RL
R
LD
WR
C
F
MR
A
BUILDABLE AREA
LAKE STRE
Select an icon.



VE BLOCK
ADJUST BLOCK
CULATION LINES
IDO CIRC. LINES
MOVE OOR PLAN
ROTATE OOR PLAN
FLIP OOR PLAN
VIEW/HIDE SITE
ART OVER
REVIEW TASK
BUILDABLE AREA
LAKE STREET
P
C
MR
R
LD
A
WR
RL
S
M
J
Select an icon.



Select an item.


DRAW COLUMNS
DRAW BEARING WALLS
DRAW BEAMS
DRAW SPACE JOISTS
CHANGE CURSOR
MEASURING TAPE
UNDO ELEMENT
START OVER
REVIEW TASK
DONE
WINDOW WALL
CHILDREN'S PATIO
ADULT STUDY AREA
ADULT STACK AREA
CHILDREN'S AREA
LENDING DESK/ LOBBY
MEETING ROOM
STAFF WORKROOM/ LOUNGE
MECHANICAL ROOM
JANITOR'S CLOSET
READING LOUNGE
ENTRANCE LOBBY
WOMEN
MEN
STORAGE
To draw a joist, click the left button where you would like the joist to begin. If you wish, select another icon.