2012-11-09

White Paper Model-Centered Analysis Process (MCAP): A Pre-Design Analysis Methodology

Andrew S. Gibbons
andy_gibbons@byu.edu

Jon S. Nelson

Robert E. Richards

Original Publication Citation

White Paper prepared for the Idaho National Engineering and Environmental Laboratory, Idaho Falls, ID

White Paper

Model-Centered Analysis Process (MCAP): A Pre-Design Analysis Methodology

by

Andrew S. Gibbons
Jon S. Nelson
Utah State University

Robert E. Richards
Idaho National Engineering and Environmental Laboratory

Introduction

This paper defines a Model-Centered Analysis Process (MCAP) for pre-design analysis (PDA) to be used in the development of instruction that is problem-based, model-centered, and situated. The methodology we describe is based on theoretical principles for analysis described in the first paper.

This methodology provides the groundwork necessary to support the future development of a designer-friendly process and tools for analysis giving average designers access to methods of analysis with more power and utility than the tools that are currently available. The new analysis methodology lends itself readily to the creation of instructional designs that are model-centered and problem-based.

The analysis methodology described is intended to be generally useful by all instructional creators (instructors, designers) regardless of specific instructional medium. We have deliberately structured the analysis process so that the analysis methodology applies to the full range of instructional applications. This includes classroom instructors teaching individual lessons, multimedia designers creating short-course products, and intelligent tutoring system

designers, particularly those situating their training in realistic performance settings using problems as a structuring principle.

Part I: Theory of Pre-Design Analysis

As outlined in the companion white paper to this one, analytic methods in the past have been invented on largely *ad hoc* bases for practical purposes. We describe here an appropriate technological theory basis for such analyses. We begin by discussing the nature of technological theory itself as distinct from scientific theory and then apply technological theory principles to generating an analysis methodology.

Technological Theory

A discussion of the theoretical grounding and implications of analysis requires a common understanding of theory: particularly of our views on the nature of technological theory. Technological theory is distinct from scientific theory (Gibbons, 1999; Merrill & Twitchel, 1993; Reigeluth, 1999). Whereas scientific theory is descriptive and used largely to explain observed phenomena, technological theory is theory of arranging structures and forces for planned intervention. It specifies intervention points within natural or artificial processes at which humans can apply force or signal in a planned way to change the future course of a process, producing a path of altered future states within an affected system.

According to Simon (1981), technological methods are formed at the interface between ideational or conceptual technologies and concrete technologies:

> *I have shown that a science of artificial phenomena is always in imminent danger of dissolving and vanishing. The peculiar properties of the artifact lie on the*

thin interface between the natural laws within and the natural laws without. What can we say about it? What is there to study besides the boundary sciences—those that govern the means and the task environment?

The artificial world is centered precisely on this interface between the outer and inner environments; it is concerned with attaining goals by adapting the former to the latter. The proper study of those who are concerned with the artificial is the way in which that adaptation of means to environments is brought about—and central to that is the process of design itself. The professional schools will reassume their professional responsibilities just to the degree that they can discover a science of design, a body of intellectually tough, analytic, partly formalizable, partly empirical, teachable doctrine about the design process. (p. 131-2)

Gibbons (1999) provides an extended discussion of this principle and its implications for instructional technology. To understand the principle of technological theory, one can think of the diversion of water into channels using baffles and barriers. The forces of an ongoing process are diverted through the application of a contrary force. The result is a redirected flow that contains some portion of the original force but now combines it with a newly applied force. Information can be used as well as force to accomplish the redirection of the process. This is the basic technological principle of the computer: the computer is essentially a technology in which information structures act on other information structures to produce new structures and forces.

Figure 1 shows how a category system for familiar technologies can be built by describing the interactions of force and information. Four basic technology classes are created as one of the two interacts with, is transformed by, or is converted into the other.

Technological theories consist of statements of principle that specify: (1) measures that can be used to identify intervention points in a process, (2) combinations of specific measured values that define specific intervention points, and (3) a description of the specific intervention(s) associated with each intervention point and projected end states for each.

An intervention consists of the application of either force or information at an intervention point: an action that injects energy or information in some way into an ongoing process. The description may also specify the instrumentality or tool through which the energy or information is applied. Thus, the statement that a baffle will be placed into the flow of water in order to redirect it implies the baffle as a tool for effecting the action.

The theoretic basis of pre-design analysis is that units identified during analysis influence, mold, and condition the instrumentalities used during instruction that actually interface energy or information with natural human processes connected with learning. The influence of energy or information from a source outside the learner does not guarantee learning will take place, but it increases the probability that it will. This "injection" of information and forces may involve both active and passive means.

The nature of the pre-design analysis artifact in part determines how an instructional product will influence learning functions. Hence the difference between analysis tools is real: each variety of analysis produces a particular type of unit that applies particular kinds of forces or particular structures of information or does these in a particular manner. The study of analysis artifact-producing issues, then, is a study in technological theory.

Theory, Artifacts, and Pre-Design Analysis

The prescriptive nature of technological theory requires that a designer know the desired

		Transformed to	
		Force	Information
Transformed from	Force	Typewriter Hammer Steel beam	Computer keyboard Microphone Pencil
	Information	Computer display Phonograph record Electronic storage media	Computer CPU Conceptual design processes Signal jamming

Figure 1. Categories of technological configuration.

goal state and invites the designer to employ consistent structuring techniques as a means of reaching it. In the companion white paper to this one, examples of existing analysis methods were compared in terms of: (1) input constructs, (2) transformation rules, and (3) output constructs.

As we now begin to describe a specific analysis methodology, we will begin from this same basis. We will describe how the output artifacts of pre-design analysis impact learning processes through instruction. Figure 2 shows the analysis process acting on a body of expertise to create an *artifact of event or content structure.* This artifact contains information in a stored form that can be used to effect subsequent transformations. In the same way a chain of chemical intermediaries during cell metabolism stores and transfers information or energy for later use.

When these transformations—which may be complex chains or relatively simple cause-effect linkages—occur, they impress information or force from the analysis artifact onto an instructional artifact. We call this transformation in Figure 2, *design processes,* and we have

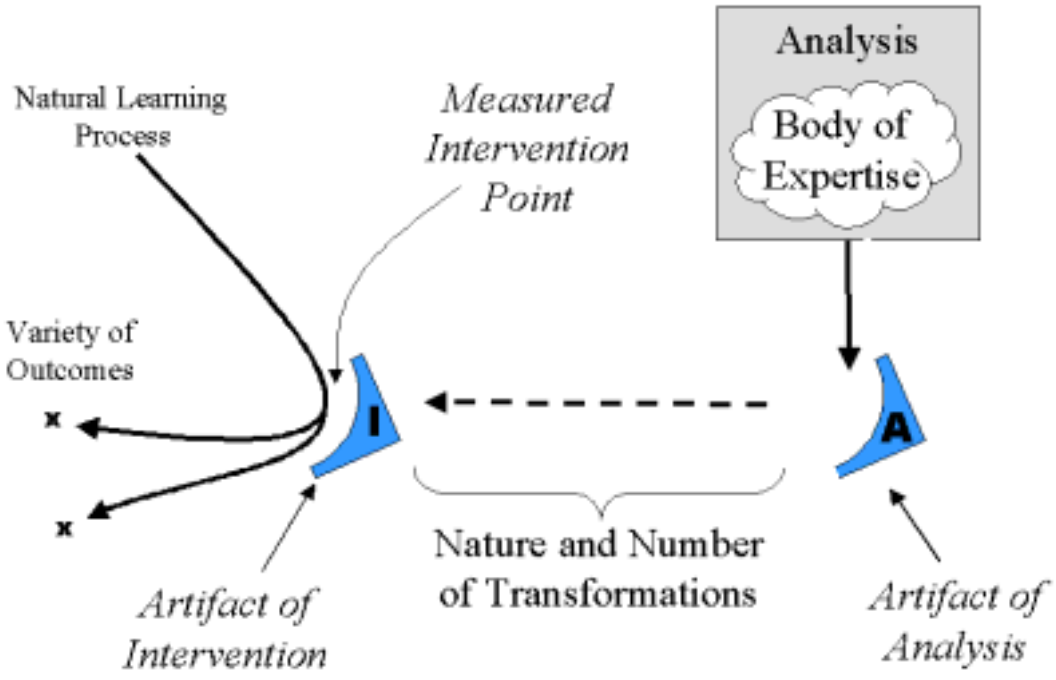

Figure 2. Technological Theory of Analysis

labeled the resulting artifact as the *artifact of intervention.* One assumption of Figure 2 is that intervention takes place at a point that has been measured as being an appropriate or perhaps optimal point for the application of that artifactual intervention.

Due to the past tendency of instructional technologists to create analysis methodologies without regard to technological theory issues of this kind, our insight into the nature of information and energy storage by intermediary analysis artifacts is limited. Likewise, our understanding of the transmission of information and energy through chaining of such artifacts is just in its infancy.

In this paper we describe a methodology that produces artifacts containing problem-event structures that can be transformed into a variety of artifacts capable of expression in a variety of forms in a variety of media, through a variety of media constructs. When these media constructs

are brought into contact with the learning processes of a student, the course of learning is influenced in some way.

As our understanding of the structures and forces that can be used in instructional communications improves, our ability to derive them from intermediary artifactual structures will also improve.

Bridging Between Analysis Theory and Design Theory

During our review of existing PDA methods (Gibbons & Nelson, 1999), significant contrasts became clear to us in the underlying principles of traditional task analysis (TTA) and more recent cognitive task analysis (CTA) methods. These differences involved seeing CTA as more than a simple extension of TTA. The differences can be summarized in terms of two properties of instruction that researchers have been trying to achieve for over four decades: *generativity* and *computability* (See, for instance, Atkinson & Wilson, 1969).

Generativity is the key to adaptive instruction. Generativity is the ability to generate or construct instructional message and interaction from primitives in response to learner responses rather than drawing whole messages from a pre-composed and stored set. The key to generativity has been the development of analysis methods that in addition to identifying task structures are capable of identifying the primitives from which instructional message and interaction are generated. The CTA methodology as it has evolved represents a group effort toward generative messaging systems. The methodology described in this paper attempts to extend CTA, particularly in the analysis of primitives that lead to interaction sequences.

Computability is the key to adaptive instruction above the message level. We define computability as the ability to generate problem structures dynamically based on recent student responses. Computability differs from generativity in the nature of the structure created.

Generativity means that an instructional system is capable of generating message and interaction. Computability means that an instructional system is capable of framing the scope within which instruction will take place. The simplest form of computability would be the dynamic selection of individual problems for a particular learner with a particular performance history from a range of pre-constructed problems. Anderson (Anderson, Boyle, Corbett, & Lewis, 1990), working with the highly structured subject-matter of mathematics, has achieved this ability in his intelligent tutors.

Gott and her associates (Gott, Lesgold, & Kane, 1996) describe how CTA contributes to the goal of computability:

> *The acquisition of complex skills occurs incrementally, in successive approximations of mature practice; therefore, the sequencing of instructional events is critical since it must promote the maturation of process. Our guiding principle regarding sequence was to decompose the targeted knowledge/skill base and reorganize it to fit learning. With most instruction the reverse is true, that is, learning is expected to conform to the way knowledge is organized in some external curriculum (or system). The fit of knowledge to learning depends on careful instructional sequencing so that the learner is always building on the foundation of prior knowledge. The cognitive task analysis findings gave us the skeleton for a learning trajectory to inform instructional sequencing. By contrasting the performances of novice, intermediate, and master technicians on problems of varying complexity (in the task analysis), it was possible to ascertain the relative learning difficulty of system components/functional areas, troubleshooting*

procedures, and strategic actions. Those findings in turn informed both the sequence of troubleshooting scenarios presented to students and the criterion performance levels to be met at each major stage along the learning trajectory. (p. 43)

As we have designed a model-centered analysis methodology we have been conscious of the goals of generativity and computability. The "views" approach which produces primitives for message construction and the use of the semantic string as a computable expression of problem structure are both means toward this end.

Instructional Theory: Problem-Basing and Model-Centering

This paper describes a methodology for PDA whose output is purposely conditioned for ready transformation into instruction that is both *problem-based and model-centered. In* this section we explain these instructional styles and present a rationale for their selection.

Problem-Based Instruction

Problem-based instruction has gained popularity in recent years, due to the widely-held view among cognitive researchers that learning itself is a problem-solving activity (Schank, Kass, & Riesbeck, 1994; VanLehn, 1993). Recently promulgated standards of the National Council for Teachers of Mathematics (National Council of Teachers of Mathematics, 1991), the American Association for the Advancement of Science (Zucker, Young, & Luczak, 1996), and others embody this trend. They urge teachers to concentrate their teaching methods in the area of problem solving activities and to refocus their curricula toward the learning of problem solving skills. In industrial training there is a similarly increasing emphasis on teaching problem solving by allowing students to solve problems.

The most influential current instructional theory, Cognitive Apprenticeship (Collins, Brown, & Newman, 1989), lists among its six prescribed instructional methods the method of Exploration that ". . . involves pushing students into a mode of problem solving on their own." (p. 483). Problem solving is supported by the other instructional methods, including observation of performance models, scaffolding of the task environment, and coaching of the performer. A method called Articulation encourages the learner to express what has been learned during problem solving and to join it to what the student already knows. During Reflection the learner learns to self-judge performances as a means of becoming independent.

Problem-based instructional methods are facilitated when an MCAP analysis is used. Precision application of problem-based instructional methods has been described by Barrows and Tamblyn (1980), and a large research literature exploring alternative methods of problem basing has developed.

Problem-basing has been selected as an assumption for the present work for two main reasons:

1. Problem-basing of instruction is highly effective when properly constructed, especially for professional-level training and training in which high stakes are involved. This finding is widespread and consistent. For this reason versions of problem-basing have been adopted in medical education (Barrows & Tamblyn, 1980), dental education (Barrows, 1998), business education (*Education Innovation in Econmics and Business (EDINEB)*, 2000), and law (Owens, 2000) over the past two decades. The current exception is to find a program of one of these types that has not adopted to some degree these methods in all or part of their curriculum. When the

University of Maastericht decided recently to open a new medical school, the design of the school included a full four-year program of problem-based instruction (Grave, 2000).

2. Practical considerations also prefer problem basing as an analysis assumption. Recent work in high-volume manufacture of technology-based (computer and web) instruction has shown that the problem has greater value as a design construct because it solves the long-standing problem of inter-media transportability of instruction. Work by Gibbons and his associates (Gibbons, Bhardwaj, & Richards, 1998) describes a "single-parse" process of development in which the problem is used as the focal point of analysis and design processes. Media-specific decisions are reserved until analysis and design are nearly complete. At that point, media-specific decisions involve the implementation of a series of problem stagings for demonstration of expert performance and learner practice. Where before three parallel design teams began early to condition instructional designs and messaging to the requirements of specific media, now a single analysis and design team builds a core of problem structures and their associated data, followed by the translation of the problems into different media. In the long term it helps the development organization standardize the training of its developers and discipline its processes, making workers effective over a wider range of instructional media.

Model-Centered Instruction

The principles of model-centered instruction (Gibbons, 1998) cause the instructional designer to think in terms of larger and more integrated design units. Traditional design

processes tend to fragment the designer's approach to the content, the instructional strategy, and the product architecture. Model-centered instruction seeks to reverse that trend. The principles of model-centered instruction, briefly stated are:

Experience. Learners should be given maximum opportunity to interact for learning purposes with one or more systems or models of systems of three types: environment, system, and/or expert performance. The terms model and simulation are not synonymous, and models can be expressed in a variety of computer-based and non-computer-based media forms.

Problem solving. Interaction with systems or models should be focused by the solution of one or more carefully selected problems, expressed in terms of the model, with solutions being performed by the learner, by a peer, or by an expert.

Denaturing. Models are necessarily denatured from the real by the medium in which they are expressed. Designers must select a level of denaturing matching the target learner's existing knowledge and goals.

Sequence. Problems should be arranged in a carefully constructed sequence for modeled (other agent) solution or for active learner solution.

Goal orientation. Problems selected should be appropriate for the attainment of specific instructional goals.

Resourcing. The learner should be given problem solving information resources, materials, and tools within a solution environment (which may exist only in the learner's mind) commensurate with instructional goals and existing levels of knowledge.

Instructional augmentation. The learner should be given support during solving in the form of dynamic, specialized, designed instructional augmentations.

Figure 3 shows several particulars of model-centering that are relevant to instruction. First, model-centered instruction represents subject-matter and expertise to the learner in the form of externalized models of real or conceptual environments, systems, and expert performances. These models may take many mediated forms, with interactive computerized models as only one possibility in a range that includes also print, audio, and visual forms of all kinds. (For instance, a model of the atom can be represented in virtually any form.) Models expressed in different media vary, however, in their information carrying capacity due to the principle of denaturing (the reduction in fidelity and/or resolution that necessarily attends any mediation).

Second, model-centered instruction uses the problem as a vehicle to focus learner interest and attention on specific relationships or complexes of relationship within models at a given time. Over time, the focus of problems enlarges to include larger portions of the target model. When one (perhaps simplified) version of externalized model is completely mastered, it expands in scope as required by the end goals of instruction and the problem focussing continues on the

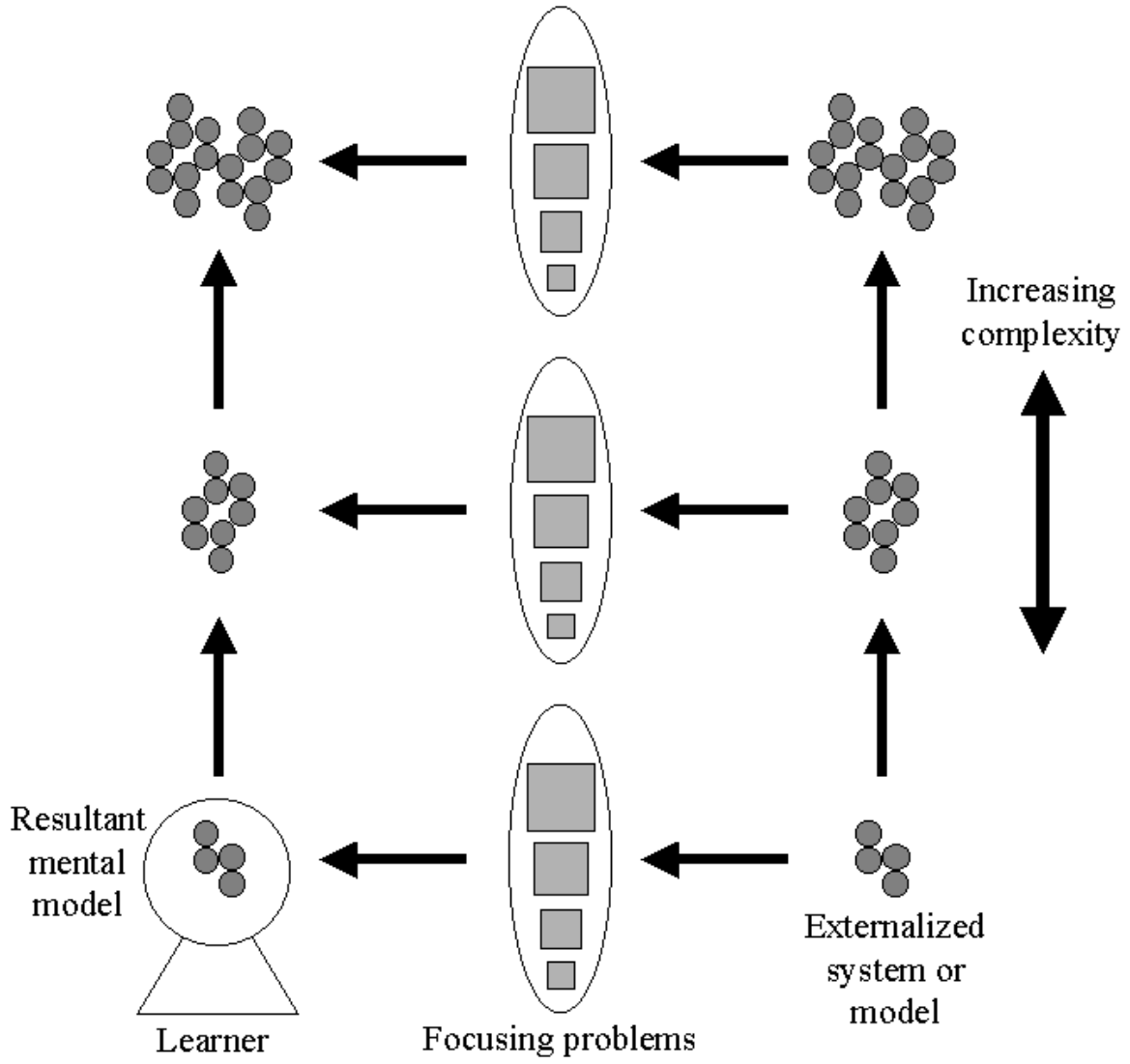

Figure 3. Model-Centered Instruction

new version. In this way, simple models of systems, environments, and skills emerge first and become more detailed and subtle over a progression of steps. Complex systems and nuanced interactions can thus be achieved systematically over time in a planned progression (See White & Frederiksen, 1990).

Model-centering has been selected as an assumption for the present work for the following reasons:

1. Model-centering encourages the designer to think in terms of reusable instructional components both for strategy and for content. The attention of the designer in traditional instructional styles is on the individual lesson, individual message patterns, individual demonstrations, and the individual practice exercise. Planning at this level frequently fails to take into account opportunities for re-use or sharing of instructional structures within and across course products. Model-centering recommends that designers work with design constructs above the horizon of the individual lesson, including shared models of environments, system and expert performance, and transportable instructional routines that adapt to new content, such as coaches (Selker, 1994).

2. Model-centering supports the designer in selecting the most appropriate models for learner interaction. Designers frequently attend to the requirements of objective-level fragmentation to the point where they lose sight of the core intentions of instruction. Those core intentions normally include experts acting on systems residing within the environment. Model-centering assumes that a key decision of the designer is the selection of these models, not the writing of fragmented instructional objectives. Objectives can be algorithmically and automatically generated from model specifications.

3. Model-centering, like problem-basing, recognizes the practical value of the problem as a design construct that crosses media barriers.

Levels of Analysis: The "Complete" Model

The MCAP analysis methodology is intended for use across a broader range of instructional designs and design projects than analysis normally supports. The central principles built into MCAP allow it to be used by individual designers who have only a day for analysis or by designers of intelligent tutors who have much longer.

The special challenge this poses is that the designer must be allowed to determine the depth of analysis and the nature of the analysis product that will be of most use for a particular project. Classroom instructors need one form and level of output, multimedia designers need another, and ITS designers need yet another. These separate levels of analysis must be based on the same principles and use the same analytic tools

Table 1 shows major dividing lines between levels of analysis determined by the nature of the project. These projects differ in the type of media they employ. More importantly they differ in the message and interaction primitives that have to be created so that corresponding media elements can be produced. This production can be carried out either at design time (pre-composed, packaged message and interaction patterns) or at the time of instruction (generated message and interaction patterns).

In analysis terms, each type of project in Table 1 requires either a different level of detail to be recorded within the common categories or detail within specialized categories, such as computable relationships between system components for full models or intelligent tutors.

This paper identifies analysis data collection sufficient for the purpose of classroom and multimedia developer needs. It identifies a major portion of the data required for full modeling. It does not identify data collection needs for intelligent tutoring systems.

Table 1. Analysis output requirements for different projects.

Type of Project	Level of Output	Level of Detail	Form of Models
Stand-up instruction	A relatively small pre-selected group of representative problems can be used for modeling demonstrations and practice within a general context of direct instructional messaging	Only detail relevant to the specific example. Some details of systems and relevant portions of the environment. These may be presented in non-graphic ways during instruction.	Slide-type, fixed illustrations of internal systems state "snapshots" that show the effects of expert performances on systems and environment.
Multimedia instruction, minimum instructor involvement	A relatively small, pre-selected group of representative problems that can be used for modeling demonstrations and practice administered by an interactive medium.	Sufficient detail to represent appearance of the controls and indicators of systems. Less emphasis on the details of the environment.	Interactive but not fully-modeled systems. In most cases developed as logic sequences that try to simulate system function with a limited number of pre-set computational paths. Environments that are represented as backgrounds for system displays but that are not necessarily realistic and sometimes may be eliminated altogether.
Full modeling environments	A wide rage of problems to be represented as degraded states of system function (for example a system	High levels of detail that can be used to build interactive models of systems. Moderate to high environment detail	Computed models of systems and perhaps environment.

	with broken components) and a full set of corresponding indications.	depending on the need to navigate environments. Performance models not strictly necessary.	
Intelligent tutoring	A wide range of problems to be represented as degraded states of system function (for example a system with broken components) and a full set of corresponding indications.	High detail in all areas: environment, systems, and expert performance.	Computed models of systems and performance. Possibly also an environmental model.

Perspective in Model Selection: The "Correct" Model

A critical phase in the methodology is the determination of what to abstract from the real world as an appropriate model or set of models for centering instruction. This phase is a process of trying to define circumscribing, the cause-effect systems that match the scope of instruction. The difficulty is that it is a subjective process conditioned by the demands of the front-end assessment, the idiosyncrasies of both the SME and the ID, and the instructional demands placed upon the SME and ID by management with respect to *what* should be instructed. Even without the front-end assessment and managerial demands, selecting an appropriate model for instruction is difficult.

An example of the difficulty can be seen in what we call the “urinalysis problem”. A group of students designing training for laboratory technicians wanted instruction to convey to technicians the process and the decisions involved in analyzing a sample of urine. They began in what seemed a logical way designing for students the control and indicator interface of the urinalysis machine. They felt this operational console was the correct model for the student to

experience through interaction. Questions arose as to the fidelity and resolution: questions like, “Should the buttons in the CBT look exactly like the buttons on the urinalysis machine?” and “How much of the functionality of the urinalysis machine should be manifest in the CBT?"

After some discussion it became apparent that the wrong model had been selected. The real goal of instruction was not to teach panel operation but the interpretation of test results produced by the machine. Whereas panel operations were trivial, interpretation of test results was a high-risk and highly complex decision-making process. What was needed was a model of the expert decision process for students to interact with. The designers were focusing on a model for use in instruction and interaction that did not correspond with the real-world performance need. After more discussion the student designers realized that the conceptual model that they started with matched the urinalysis machine--the physical cause-effect system—perfectly while slighting the expert’s performance cause-effect model—one that captured the judgment and decision making necessary to process a urine sample.

The final result was an interface that bore little resemblance to the actual urinalysis machine. The interface provided the same type of results that the machine did, but it was organized in a way that facilitated an understanding of the expert decision process and interaction with an expert.

Part II: The MCAP Methodology of Analysis

The Resonant Structure

Figure 4 shows that the output of the MCAP methodology is a design element—the problem structure—and that this element is related to three classes of analytic element: environment elements, cause-effect system elements, and elements of expert performance. The

arrows in Figure 4 show relationships that create a property we call *resonance.* The principle of resonance is that any type element of the analysis may be used as an entry point for the systematic derivation of the remaining elements of the other types. For instance, the identification of an environment element leads directly to the identification of system process elements, related expert performance elements, and eventually to problems that involve all of these. Likewise, the identification of a problem allows the designer to work backward to define the environment, system, and expert performance requirements necessary to stage that problem for students. The basic unit of MCAP analysis is this resonant structure.

This resonant relationship exists for all four of the Figure 4 elements in all of the directions indicated by arrows. The implication is that analysis does not necessarily proceed in a top-down manner as is true in most analysis methodologies but that the analyst may move laterally among design elements in a pattern more compatible with a subject-matter expert's stream of thought. We believe that even traditional forms of analysis proceed more or less in this fashion, even during analyses that are putatively "top-down". The analysis begins at some initial anchor point and works outward in all directions, sometimes working upward to a new anchor.

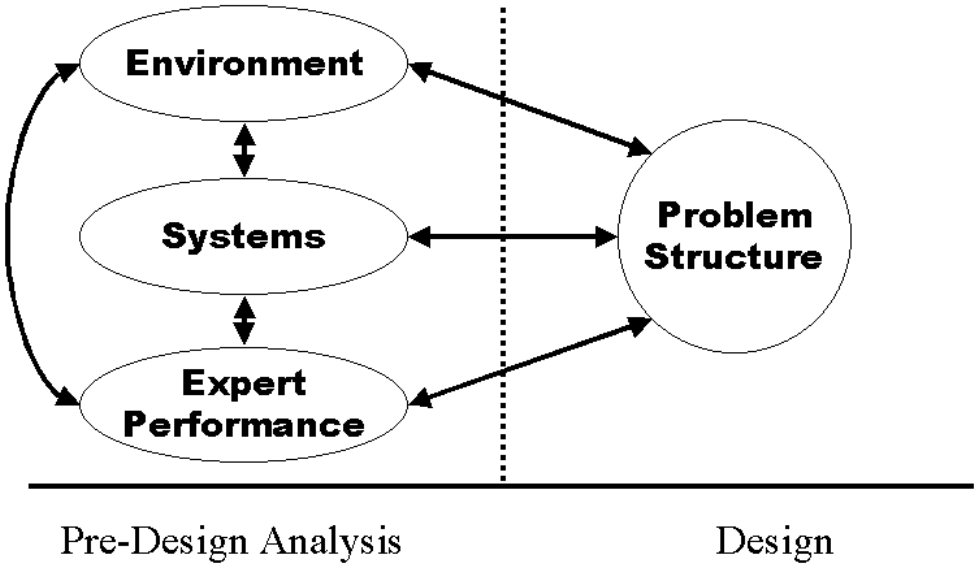

Figure 4. The Resonant Structure

Figure 5 shows that each of the element types from Figure 4 participates in a hierarchy of elements of its own kind. These hierarchies can be projected, as it were, on the views of a modeling language. This modeling language, which we have termed an Analysis Modeling Language (AML), is patterned after the Unified Modeling Language (UML) used by programmers to design complex program systems (Booch, Rumbaugh, & Jacobsen, 1999).

This modeling language offers four projected views of a body of expertise: a view of performance environments, a view of cause-effect systems hosted within the environments, and expert performances performed on the cause-effect systems within the environments. The fourth

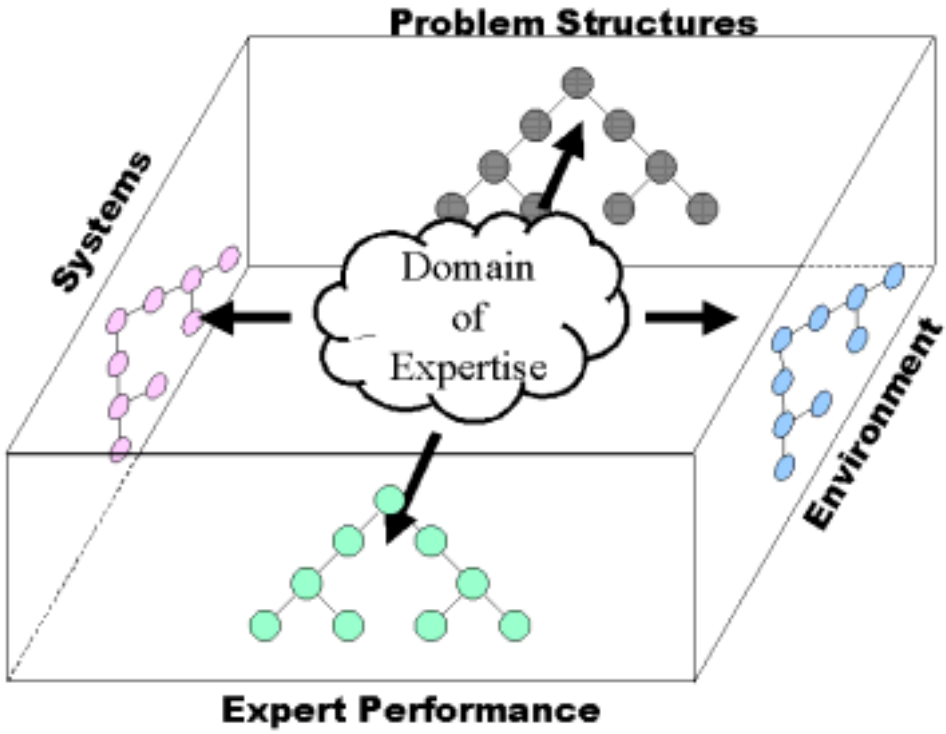

Figure 5. Analysis Modeling Language (AML) views

view into the body of expertise consists of situated problem structures from everyday settings that can be used for instructional design purposes. Problems in the problem view are linked with the elements from the other views in resonant relationships. This linkage is shown in Figure 5 by a dotted line and in Figure 4 as the arrows connecting analysis and design elements. The dotted line is an oversimplification intended to relate Figures 4 and 5 and represents relationships that in reality are more complex than shown.

The benefit of representing the analysis as a set of views linked internally is that the relationships between elements within a view are preserved and can be used to further the analysis. The principle of resonance allows the analyst to move between views, filling in the hierarchy on each of the views. The analyst is also enabled to work within a single view,

generating upward and downward from individual elements according to the logic of that individual hierarchy.

For instance, an analyst, having defined a system process, may break the process into its sub-processes showing them hierarchically on the same view and then move to a different view, say the expert performance view, to identify tasks related to the control or use of the sub-processes that were identified in the first view. This may in turn suggest appropriate training problems to the analyst, so the analyst may move to the problem view and record these problems.

The Organization of the Views

The hierarchies of each view differ according to a logic unique to that view:

- The *environment view* hierarchy breaks the environment into locations that can be navigated by paths. Environment locations are normally nested within each other, and diagrams are often the best representation of their interrelation. However, a simple outline form can capture this relationship also. Paths between locations must be captured in a supplemental form when an outline is used.

- The *system view* contains three hierarchies under a single head: (1) a raw component hierarchy, (2) a functional subsystems hierarchy, and (3) a system process hierarchy. Examples of these relationships include: (1) the division of an automobile engine into physical components determined by proximity or juxtaposition, (2) the division of an automobile engine into sometimes physically isolated parts that form functional subsystems, such as fuel system, and cooling system, and (3) a separate hierarchy describing processes carried out as forces and information operate and are transformed within the system defined by (1) and (2). The system view in most cases

will also include a view of the product produced by expert performance and/or the tools used to produce the product.

- The *expert performance view* decomposes complex, multi-step performances into progressively simpler performance units according to a parts-of or varieties-of principle. Several systems for CTA have been developed that perform this kind of breakdown. Moreover, TTA accomplishes this type of a breakdown but to a lesser degree of detail and without including key decision making steps.

- The *problem structure view* contains a hierarchy of problem structures systematically derivable from the contents of the other views using the parameterized semantic string as a generating device (see description below). This view arranges problems in a multi-dimensional space according to field values in the string structure. As strings take on more specific modifiers they move downward in the hierarchy.

The environment, system, and expert performance views are composed of analytic elements. The problem structure view is composed of design (synthesized) elements that have an analytic function, hence the connection of the problem view to the other three. This makes the set of views, taken together, a bridge between analysis and design.

Entering Analysis from Multiple Points

The principle of resonance allows for multiple entry points into the analysis. The analyst can begin by collecting environment elements, system elements, elements of expert performance, or problem structure elements and organizing them into views, and once information is gathered

for one analysis view, resonance automatically leads the designer to questions that populate each of the other views.

Problem Structures. Analysis can begin with a set of constructs normally considered to be on the design side of the analysis-design watershed. This view of analysis means that, as analysts we can begin by asking the SME what they think are appropriate performance problems (job situations, common crises, use cases, etc.) for instruction as a means of moving analysis ahead using constructs from the SME's world that are already familiar.

As a SME begins to generate examples of performance problems, the ID also must translate the statements into a semantic string form, either at the time of analysis or in a follow-up documentation period. The ID also must use the resonant relationships principle to identify elements of performance, systems, and environment implicit within problem statements and record them in their respective views. Additional problems can be generated from initial problems by formalizing problem statements into semantic string form and systematically varying string slot contents to create new problem forms.

Expert Performance Structures. Currently there exists a number of tools for both elicitation and recording of expert performance. This area has been the special focus of analysis in the past for both TTA and CTA. TTA has tended to proceed by fragmenting a higher-level task into lower-level components. CTA has tended to look for sequences of tasks, including reasoning and decision-making steps—especially those related to specific characteristics of the operated system. Performance analysis in MCAP incorporates both of these principles, with emphasis on the hierarchical arrangement of tasks because of the generative principle it establishes for continuing analysis using existing tasks to generate new ones.

To expedite analysis with the SME, a *use case* approach is appropriate for identifying both task fragments and the decisions that join them into longer sequences of performance. A sufficient number of use cases gathered quickly can provide the analyst with a great deal of analysis detail, and in cases of restricted development time can provide a rapid analysis alternative because use cases constitute a basis for problem sets.

Environment Structures. An environment is a system that is not within the immediate scope of instruction. In instruction that uses progressions of models as a method (White & Frederiksen, 1990), what is initially environment eventually emerges into the details of the systems being instructed. Therefore, environment is a relative and dynamic construct. If a particular system is not at the forefront of instruction, in the context of a specific problem, it can be considered the environment or the background for the problem. Environment provides both setting elements and pathing elements for the processes described in the system view of MCAP. An environment description can be quite detailed, and most SMEs tend to accept this as a standard. However, Lesgold (1999) and Kieras (1988) have recommended that both environment and system definitions need to be limited to useful definitions from the student's point of view to avoid including irrelevant, unusable information in instruction.

A good starting point for eliciting elements of the environment is to ask the SME for all of the settings where systems exist or performances are required. One way of capturing the environment is as a diagram using AML. Representing an environment graphically helps both SME and ID to ensure completeness in the environment view and to use the environment view to extend other views by path tracing.

System Structures. Understanding the processes within a system is a prerequisite to explaining behavior and outcomes with respect to that system. A significant source of operator

error is the lack of a complete and accurate system model in the learner. It is clear that good system models are the basis for effective expert performance and that as expertise grows the nature of the expert's system models changes correspondingly (Chi, Glaser, & Farr, 1988; Psotka, Massey, & Mutter, 1988). From our review of the literature we found a number of instructional products that did not succeed as well as they could have because they lacked system process that could be separately articulated. MYCIN (Clancey, 1984), for instance, could not give explanations of expert systems decisions without system models. Instruction that can convey to the learner a complete model the processes that occur within the scope of instruction can provide the learner with a complete explanation of why certain phenomena were observed.

In system process analysis three things must be identified: initiating events, internal processes, and terminating indications. Events that initiate a system process consist of a user action or another process acting from without. Internal processes are represented in a number of ways: as sequential steps, as flow diagrams, or as principles (rules) that control the flow of events.

System structures are captured in the form of: (1) a hierarchy of system components, (2) a hierarchy of functional units made up of individual components, and (3) a tracing of the processes on the face of (1) and (2) on top of the environment decription. Process tracings form a multi-dimensional hierarchical form but are best captured as individual tracings, normally related to expert performance elements.

The Semantic String as a Construct for Problem Structure Expression

The output of MCAP is a set of problem structures (with their resonant environment, system, and expert performance primitives) that can be used to build an instructional curriculum sequence. A problem structure is a complete and detailed task description expressing a

performance to be used during instruction, either as an occasion for modeling expert behavior or as a performance challenge to the learner.

The MCAP problem structure is a data structure. A repeating data structure of some kind is common to all analysis methodologies. This is most evident in TTA in the repeating nature of tasks at different levels of the hierarchy and in CTA in the PARI unit (Hall, Gott, & Pokorny, 1995), the regularity of Anderson's rule forms (Anderson, 1993), and the regular analysis structures by the DNA and SMART (Shute, in press) systems. It is likely that the regularity of these analysis units is closely related to a conceptual unit defined by Miller, Galanter, and Pribram (Miller, Galanter, & Pribram, 1960) called the TOTE (Test-Operate-Test-Evaluate) unit.

The MCAP problem structure is expressed as a semantic string—created by merging data fields from the other three analysis views: (1) environment, (2) cause-effect systems, and (3) expert performance. The semantic string expresses a generic problem structure. During instruction a problem structure is given specific instantiating values. The semantic string does not have an absolute structure and can therefore be adapted to the characteristics of tasks related to individual projects and to trajectories of student progress. However, we believe the string to be conditioned by a general pattern of relationships found in everyday event-script or schematic situations (Schank et al., 1994) in which actors act upon patient systems and materials using tools to create artifacts. We believe this dramatic structure to be related to Schank's (Schank & Fano, 1992) list of indices.

A general expression of the semantic string consists of the following:

In <environment> one or more <actor> executes <performance> using <tool> affecting <system process> to produce <artifact> having <qualities>.

This general expression of the semantic string can, in turn, be broken down into more detailed parts corresponding to the detailed definition of the environment, of the cause-effect systems, and of the performance.

The general environment portion of the string can be expressed as follows:

In <location> of <environment> in the presence of <external conditions> in the presence of <tool> in the presence of <information resources> in the presence of <material resources>.

The general system portion of the string can be expressed as follows:

Affecting <system> that exists in <state> manifest through <indicator> and operated using <control>.

The general performance portion of the string can be expressed as follows:

One or more <actor(s)> execute <performance> using <method> with <technique>.

Benefits of the Semantic String

One of the functions of analysis is accountability. Analysis becomes a part of the process of *requirements tracing* (Jarke, 1998) for instructional purposes. Designers must be able to

demonstrate that they have achieved some degree of coverage of some body of subject-matter with their instruction.

Accountability requirements have traditionally led to forms of instruction that fill administrative requirements but have little impact on performance. This is especially true when training is regulated and mandated (aviation, nuclear, power distribution, hazardous waste). Accountability in these cases has been equated with verbal coverage, and a formulaic variety of verbal training has become standard in these situations (*Guidelines for Evaluation of Nuclear Facility Training Programs*, 1994).

Instructional objectives are normally used as the accountability tool in forming this type of instruction, and in some cases traditional task analysis methods are used as a means of grounding the objectives in a systematic process to certify soundness and completeness. Accountability in this atmosphere is difficult, and sometimes task analysis principles have to be stretched in order to make the accountability connection.

Acceptance of problem solving as appropriate form of instruction and assessment makes the accountability problem harder. It creates new problems for accountability, because the basic construct of accountability changes from the verbal check-off to the real and dynamic competency. Instructional designers lack the ability to express dynamic competency and also lack a theory of performance measurement that would generate appropriate performance assessments.

The semantic string mechanism supplies a method for the description of dynamic competency. When the string is instantiated with specific values or with a range of values, it expresses a specific problem or range of problems. Variations of string values make this an expression of a range of performance capability.

Generating Problems and Using Weighting To Focus Problem Sets

Instructional problems are generated computationally using the semantic string by defining a range of values for each field in the string and then systematically substituting values in specific string positions. Generation of problems using the semantic string takes place in two steps: (1) insertion of values from the hierarchically-organized views into the string to create a problem, and (2) selection of specific initial values that instantiate the problem. This results in a geometric proliferation of possible problems, so mechanisms capable of narrowing and focusing problem sets into sequences are important.

This is accomplished by selecting string values depending on the principle the designer is trying to maximize within a problem sequence. A few possible sequence principles are given here as examples:

- *Maximum coverage in limited time*—String values will be selected with the minimum of redundancy. Each problem will contain as many new elements in string positions as possible.
- *Cognitive load management*—String values will be selected in terms of their addition to the current cognitive *load*. Increases may be due to increased memory requirement, coordination of conflicting sensory demands, integration of parallel decision processes, or a large number of other possibilities. Each string element is judged according to its contribution to load.
- *Integration of complexes of prior learning*—String values are selected as combinations of elements from each of the view hierarchies that practice already mastered areas of the hierarchies in new combinations.

- *Decontextualization of skills*—String values are selected so that they vary systematically, preserving expert performance elements but varying environment and system elements as widely as possible. Core performances are retained in the string but to them are added as wide a variety as possible of non-related performances.
- *Practice to automaticity*—String values are kept as unchanged as possible with the exception of the conditions in the environment, which change in terms of timing factors where possible.
- *Transfer*—String values for expert performance change along a dimension in which performances in the sequence contain similar elements. Environment and system string elements
- *Risk awareness*—String values are selected on the basis of weightings attached to performances, system processes, and environmental configurations that have historically or have the potential for posing risks.

When string values have been selected, individual problems are instantiated by the designer by specifying data that situates the problem. This data includes:

- *Environment configuration data*—Data that describes the specific environment in which the problem will be presented to the learner.
- *Environment initialization data*—Data that describes variable values of the environment at problem initiation.
- *System configuration data*—Data that describes the configuration of systems that the interact with or see demonstrations on.
- *System initialization data*—Data that describes variable values of the systems at the beginning of the problem.

- *Problem history*—Data that describes the history of events that has brought the problem sent state.
- *Problem end state data*—Data that describes the states of system and environment at the successfully concluded problem.

Roles of Instructional Designer and Subject-Matter Expert

The previous discussion of learning objectives and MCAP highlights a qualitative change in the role that learning objectives play in instruction. At the same time a new and different role is required of both instructional designers and subject-matter experts. For the instructional designer their focus changes from simply trying to produce something that satisfies the instructional objectives to trying to articulate a problem solving process. Hopefully this will be more intuitive and rewarding for the instructional designer.

At the same time, because the objectives evolve in a natural form, the analysis process prescribed by MCAP more readily connects with the understanding of the SME. It is not necessary to focus on "objectives" as a starting point. This has always been a point of difficulty since SMEs do not generally deal with objectives. When using design jargon like "objectives" or "task analysis" with a SME you can quickly lose them and their interest in the project. When using the MCAP methodology the analysis process itself is facilitated because the SME responds in terms of problems, scenarios, and use cases with which the SME is familiar from daily involvement. This enables the SME to participate fully and effectively from the beginning of the analysis process—avoiding learning new terminologies and using familiar patterns of thinking about performances.

Conclusion

The pre-design analysis process (MCAP) methodology described here provides the groundwork necessary for future work in the disciplined development of explicit procedures and tools for PDA. At the heart of this methodology are two constructs that provide the rigor we require of an analysis methodology: the resonant structure and the semantic string. These constructs provide rigor because they are designed to (1) produce model-centered, problem-based structures and (2) they provide generativity and computability. Model-centering and problem-basing is evident in that each construct is a tool for gathering and organizing as well as for representing expert behavior, system information, environment information, and problem structure information. Generativity and computability are evident in that each construct identifies the primitives from which instructional messages and interactions can be generated and accounts for problem sequencing.

Having broken this important ground, further amplification to the methodology is needed in the development of detailed heuristics and procedures. These methods and procedures could then be illustrated and tested with examples from a variety of training needs.

References

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. W. (1990). Cognitive Modeling and Intelligent Tutoring. *Artificial Intelligence, 42*, 7-49.

Atkinson, R. C., & Wilson, H. A. (Eds.). (1969). *Computer-Assisted Instruction: A Book of Readings*. New York: Academic Press, Inc.

Barrows, H. S. (1998). The Essentials of Problem-Based Learning. *Journal of Dental Education, 62*(9), 630-633.

Barrows, H. S., & Tamblyn, R. M. (1980). *Problem-Based Learning: An Approach to Medical Education*. New York: Springer.

Booch, G., Rumbaugh, J., & Jacobsen, I. (1999). *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley.

Chi, M., Glaser, R., & Farr, M. (1988). *The Nature of Expertise*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Clancey, W. J. (1984). Extensions to Rules for Explanation and Tutoring. In B. G. Buchanan & E. H. Shortliffe (Eds.), *Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Mathematics. In L. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Education Innovation in Econmics and Business (EDINEB)(2000)., [web page]. Available: http://www2.unimaas.nl/~edineb/.

Gibbons, A. S. (1998). Model-centered instruction, *Annual meeting of the american educational research association*. San Diego, CA.

Gibbons, A. S. (1999). *The Practice of Instructional Technology*.Unpublished manuscript, Logan, UT.

Gibbons, A. S., Bhardwaj, K. K., & Richards, R. (1998). The Single-Parse Method of Design for Problem-Based Instruction. *Educational Technology Research and Development, 46*(2), 110-116.

Gibbons, A. S., & Nelson, J. S. (1999). *Theoretical and Practical Requirements for a System of Pre-Design Analysis*. Idaho Falls, ID: Idaho National Engineering and Environmental Laboratory.

Gott, S. P., Lesgold, A. M., & Kane, R. S. (1996). Tutoring for Transfer of Technical Competence. In B. G. Wilson (Ed.), *Constructivist learning environments: Case studies in instructional design* (pp. 252). Englewood Cliffs, NJ: Educational Technology Publications.

Grave, W. (2000). *Maastricht Problem-Based Learning Site*, [web page]. Available: http://www.unimaas.nl/pbl/ [2000.

Guidelines for Evaluation of Nuclear Facility Training Programs (DOE-STD-1070-94)(1994). Washington, D.C.: U.S. Department of Energy.

Hall, E. P., Gott, S. P., & Pokorny, R. A. (1995). *Procedural Guide to Cognitive Task Analysis: The PARI Methodology* (AL/HR-TR-1995-0108). Brooks, AFB, TX: Armstrong Laboratory, Human Resource Directorate.

Jarke, M. (1998). Requirements Tracing. *Communications of the ACM, 41*(2).

Kieras, D. E. (1988). What Mental Models Should Be Taught: Choosing Instructional Content for Complex Engineered Systems. In J. Psotka & D. L. Massey & S. A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned* (pp. 85-111). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A. M. (1999). Intelligent Learning Environments for Technical Training: Lessons learned. In A. K. Noor (Ed.), *Workshop on Advanced Training Technologies and Learning Environments*. Hampton, VA: NASA Langley Research Center.

Merrill, M. D., & Twitchel, D. G. (Eds.). (1993). *Instructional Design Theory*. Englewood Cliffs, NJ: Educational Technology Publications.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the Structure of Behvior*. New York: Henry Holt and Company, Inc.

National Council of Teachers of Mathematics. (1991). *Professional Sstandards for Teaching Mathematics / Prepared by the Working Groups of the Commission on Teaching Standards for School Mathematics of the National Council of Teachers of Mathematics.* Reston, VA: The Council.

Owens, R. (2000). *"Self Teaching" groups in Constitutional Law*, [web page]. Available: http://web.acue.adelaide.edu.au/leap/focus/pbl/owens.html [2000.

Psotka, J., Massey, D. L., & Mutter, S. A. (Eds.). (1988). *Intelligent Tutoring Systems: Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Reigeluth, C. M. (Ed.). (1999). *Instructional-Design Theories and Models: An Overview of their Current Status*. Mahwah, NJ: Lawrence Erlbaum Associates.

Schank, R. C., & Fano, A. (1992). *A Thematic Hierarchy for Indexing Stories*. Evanston, IL: The Institute for the Learning Sciences, Northwestern University.

Schank, R. C., Kass, A., & Riesbeck, C. K. (Eds.). (1994). *Inside Case-Based Explanation* (Vol. 3). Hillsdale, NJ: Lawrence Erlbaum Associates.

Selker, T. (1994). COACH: A Teaching Agent That Learns. *Communications of the ACM, 37*(7).

Shute, V. J. (in press). DNA: Towards an Automated Knowledge Elicitation and Organization Tool. In S. P. Lajoie (Ed.), *Computers as Cognitive Tools: No More Walls* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Simon, H. A. (1981). *The Sciences of the Artificial* (2nd ed.). Cambridge, MA: MIT Press.

VanLehn, K. (1993). Problem Solving and Cognitive Skill Acquisition. In M. I. Posner (Ed.), *Foundations of Cognitive Science*. Cambridge, MA: MIT Press.

White, B. Y., & Frederiksen, J. R. (1990). Causal Model Progressions as Foundations for Intelligent Learning Environments. *Artificial Intelligence, 42*, 99-157.

Zucker, A. A., Young, V. M., & Luczak, J. M. (1996). *Evaluation of the American Association for the Advancement of Science's Project 2061*. Menlo Park, CA: SRI International.