Representing Instructional Material for Scenario-Based Guided-Discovery Courseware

Frank L. Greitzer
Pacific Northwest National Laboratory
Richland, WA 99352
frank.greitzer@pnl.gov

M. David Merrill
Brigham Young University
Laie, Hawaii 96762
merrill@byu.edu

Douglas M. Rice, Darren S. Curtis
Pacific Northwest National Laboratory
Richland, WA 99352
doug.rice@pnl.gov; darren.curtis@pnl.gov

ABSTRACT

The focus of this paper is to discuss paradigms for learning that are based on sound principles of human learning and cognition, and to discuss technical challenges that must be overcome in achieving this research goal through instructional system design (ISD) approaches that are cost-effective as well as conformant with today's interactive multimedia instruction standards. Fundamental concepts are to: engage learners to solve real-world problems (progress from simple to complex); relate material to previous experience; demonstrate what is to be learned using interactive, problem-centered activities rather than passive exposure to material; require learners to use their new knowledge to solve problems that demonstrate their knowledge in a relevant applied setting; and guide the learner with feedback and coaching early, then gradually withdraw this support as learning progresses. Many of these principles have been put into practice by employing interactive learning objects as re-usable components of larger, more integrated exercises. A challenge is to make even more extensive use of interactive, scenario-based activities within a guided-discovery framework. Because the design and construction of interactive, scenario-based learning objects and more complex integrated exercises is labor-intensive, this paper explores the use of interactive learning objects and associated representation schema for instructional content to facilitate development of tools for creating scenario-based, guided-discovery courseware.

ABOUT THE AUTHORS

Frank L. Greitzer has thirty years of R&D experience in areas of cognitive psychology, user-centered design, expert systems, and advanced performance support and training. After receiving a BS in mathematics in 1968 from Harvey Mudd College and a PhD in mathematical psychology in 1975 from the University of California, Los Angeles, Dr. Greitzer held university teaching positions and R&D positions in the federal government and the aerospace industry prior to joining the Pacific Northwest National Laboratory (PNNL) in 1992. As a principal scientist at PNNL, Dr. Greitzer has managed a variety of R&D programs, including advanced distributed learning and training applications. Dr. Greitzer's current research focus is on advanced training technology and applications of cognitive science to support information analysis.

M. David Merrill is a Professor Emeritus, Utah State University and Professor, Brigham Young University, Hawaii. Since receiving his Ph.D. in 1964 his primary interest has been investigating what makes for effective, efficient and appealing instruction. In the pursuit of this goal he has made several major contributions to the field of instructional technology. In the 1970s he was a primary designer of the authoring system for TICCIT, an early CBT system. In the 1980s he developed Component Display Theory and Elaboration Theory that are still widely used as guides for effective instructional development. In the 1990s his work included knowledge objects and instructional transaction theory primarily aimed at facilitating the automation of instructional design. His recent work is an attempt to identify those first principles of instruction that are common to most theories and models of instruction. He was recently awarded the 2001 Distinguished Service Award by AECT "for advancing the field of Instructional Technology through scholarship, teaching, and leadership."

Douglas M. Rice received degrees in Computer Science and Education at Washington State University. Since joining PNNL in 1994, he has focused on Internet application development, distance learning, and collaboration research and development. Mr. Rice is the technical lead and primary code developer of Pachelbel, a distributed learning development/delivery system that has been used to develop over 20 innovative distributed learning applications for a variety of US government and commercial clients, as well as for Web-based training and information technology products at PNNL.

Darren S. Curtis received a BS in computer science from the University of Idaho and a MS in computer science from Washington State University. A senior research scientist at PNNL with seventeen years of computer technology R&D experience, Mr. Curtis has served as lead system architect and technical contributor on many computer technology applications involving large scale UNIX systems deployment and management; large scale data collection, storage, and distribution; and large scale networking applications. Mr. Curtis also has developed award-winning Web-based distributed learning applications.

Representing Instructional Material for Scenario-Based Guided-Discovery Courseware

Frank L. Greitzer
Pacific Northwest National Laboratory
Richland, WA 99352
frank.greitzer@pnl.gov

M. David Merrill
Brigham Young University
Laie, Hawaii 96762
merrill@byu.edu

Douglas M. Rice, Darren S. Curtis
Pacific Northwest National Laboratory
Richland, WA 99352
doug.rice@pnl.gov; darren.curtis@pnl.gov

INTRODUCTION AND BACKGROUND

The focus of this paper is both practical and technical. Our research aim is to develop and apply paradigms for learning that are effective, based on sound principles of human learning and cognition, and that will fit within current standards and practices for instructional design and interactive multimedia instruction (IMI). There are implementation challenges (technical issues) that must be overcome in making this endeavor cost-effective as well as conformant with today's IMI standards. The goal of this paper is not to offer complete solutions to these problems, but rather to identify the problems and describe a possible technical approach to solving the problems.

Merrill (2002a) and Greitzer (2002) have each argued for an approach to instructional systems design (ISD) that is based on first principles of instruction, grounded in research in learning and cognition. Merrill (2002a) observes that the most effective learning environments are those that are problem-based and involve the student in four distinct phases of learning: (1) activation of prior experience, (2) demonstration of skills, (3) application of skills, and (4) integration of skills into real-world activities.

The first phase, activation of prior experience, is based on a solid foundation of behavioral research that has shown that learning and memory are facilitated when the learner is able to relate the new knowledge or skills to existing knowledge. The cognitive-based approach to IMI (or e-Learning) described by Greitzer (2002) builds upon this notion in constructing an instructional framework that exploits human associative and organizational processes to stimulate semantic memory and to build understanding of new, more complex concepts from more basic ones.

The second phase, demonstration of skills, consists of presenting or demonstrating information to the learner. Merrill (2001) describes the demonstration/presentation phase as comprising two components: TELL, referring to the instructional method of presenting general information to the student (telling about a definition, steps of a procedure, etc.); and SHOW, referring to the instructional method of presenting or demonstrating specific information, such as an instance of a concept or a demonstration of a procedure.

The third phase, application of skills, generally refers to practice. Merrill (2001) describes two components of the application/practice phase, ASK and DO. ASK refers to requiring the learner to recall information that was presented; DO refers to requiring the learner to *apply* the newly gained knowledge. It is not sufficient, therefore, simply to ask the learner to recall a concept or a definition. Learning is facilitated when learners are required to use their new knowledge or skill to solve problems.

The fourth phase, integration of skills, refers to the ability to transfer the new knowledge or skill into one's everyday life. In other words, the new knowledge and skills should have real-world *relevance*. Merrill (2002a) observes: "The real motivation for learners is learning. When learners are able to demonstrate improvement in skill, they are motivated to perform even better…. Learners need the opportunity to reflect on, defend, and share what they have learned if it is to become part of their available repertoire."

Unfortunately, it is all too common that instructional strategies focus primarily on only one or two of these instructional phases and ignore the other phases that are critical to effective learning. Thus, historically and up to the current time, there is a tendency for instructional systems to focus on demonstration of skills (especially the TELL component of demonstration) and on application of skills (especially the ASK component of practice). As Merrill (2001) points out, "Too much current instruction is TELL & ASK when it would be more effective if it included SHOW and DO. Some recent approaches to instructional design emphasize DO but neglect to include adequate SHOW."

These cognitive based instructional design principles imply the following guidelines for effective learning systems: (a) relate material to previous experience (learner uses relevant experience as a foundation for new knowledge); (b) demonstrate what is to be learned using interactive, problem-centered activities rather than passively telling about the material; (c) engage learners by requiring them to use their new knowledge to solve real-world problems; (d) progress from simple to complex problems tasks, guiding the learner with feedback and coaching early and gradually withdrawing this support as learning progresses; and (e) encourage learners to use the new knowledge or skill in everyday life (demonstrate knowledge in a relevant applied setting).

The way in which an instructional system meets these guidelines is influenced by the way the instructional designer structures the training material. Clark (1998) suggested four instructional architectures that reflect different assumptions about how learning occurs, the role of the instructor or instruction, and the final goal of the instruction:

- *Receptive*: The learner is like a sponge that absorbs the instructional material.
- *Directive*: The instruction sequences and chunks the material and provides frequent opportunities for learners to respond, and corrective feedback is provided.
- *Guided Discovery*: The instruction provides learners with problems adapted from actual work settings; the instructional system facilitates acquisition of knowledge and skills through providing the learners with experience on these problems.
- *Exploratory*: The learner has maximum control in navigating through the instructional material that comprises information, examples, demonstrations and exercises.

It is evident that as the architecture progresses from Receptive to Exploratory, the prescribed role of the learner changes from passive/constrained to active/unconstrained. Many traditional courses adopt the receptive or directive architectures: information is presented in a series of lessons, each of which is followed by some multiple-choice or objective questions to test the learner's understanding. In the Guided Discovery architecture, the goal is to construct a more experiential approach that presents realistic problems (also called scenarios) and to provide coaching to facilitate learning. As the learner gains knowledge and skill, the level of coaching diminishes (the "scaffolding" is gradually withdrawn) and more responsibility is left to the student[1]. The Exploratory architecture is open-ended from the outset—i.e., no scaffolding (coaching) is provided and the learner is left to his or her own devices to acquire the knowledge and skills that meet learning objectives (hence it is sometimes called "sink or swim" courseware).

We focus on guided discovery because we believe that this instructional architecture, more than the others, offers the greatest potential for cognitive skills training. This view is consistent with a number of extant instructional theories including Nelson's (1999) collaborative problem solving guidelines, Jonassen's (1999) constructivist learning environments, van Merriënboer's (1997) four component instructional design model, and Schank, Berman, & MacPerson's (1999) learning by doing model. In the remainder of this paper, we explore more fully the nature, the development challenges, and the implementation concepts of employing authentic learning tasks within a guided-discovery approach to instruction.

CHALLENGES

Cognitive Load

One of the risks of using realistic, rich learning tasks is that learners may become overwhelmed by the complexity of the tasks. This is the problem of cognitive load—the training challenge is how to manage cognitive load when realistic tasks are used even in the initial stages of learning. Cognitive load theory (van Merriënboer, Kirschner, & Kester, 2003) offers guidelines for decreasing cognitive load on the learner to enable learning to occur. Providing performance support (scaffolding) enables a learner to achieve a goal initially; when the learner achieves the initial goal, support is gradually diminished until it is no longer needed. Cognitive load theory emphasizes the need to integrate support for novice learners with the task environment so the learner does not have to deal mentally with the added cognitive load of the support system itself.

One method of providing scaffolding is to progress from simple to complex tasks. Complex tasks may be

[1] This approach is comparable to the constructivist approach (Jonassen, 1999) that emphasizes problem solving, starting with tasks that learners know how to perform, gradually adding task difficulty until they are unable to perform alone, providing demonstrations such as worked examples and coaching (scaffolding) to help the learner understand the desired performance, and gradually removing the scaffolding as knowledge and skills are acquired.

broken down into simpler parts that may be trained separately and gradually combined into whole tasks. Using this approach, it may not be until the end of the training course that learners have an opportunity to perform the whole task. This approach was demonstrated by Greitzer and colleagues (Greitzer, 2002; Greitzer, Rice, Eaton, Perkins, Scott, Burnette, & Robertson, 2003) in an IMI application to train soldiers to operate and maintain logistics communication equipment. Smaller instructional elements (called interaction elements within an IMI framework) were defined to correspond to concepts, skill, or procedures to be learned. These elements were used in earlier stages of learning and later combined into more complex objects to meet broader learning objectives. This was designed to limit the learner's cognitive load by activating, demonstrating, and applying knowledge associated with the more elementary concepts early in the course, then exposing the learners to more complex combinations of these concepts only after they have acquired the knowledge and skills for the components. At the end of the course, the learner was required to demonstrate a more holistic understanding by passing a final, integrated exercise that was constructed from the simpler components. In this way, the cognitive load associated with the whole task is reduced compared to what would be expected without the part-task training.

While part-task approaches help to manage the learner's cognitive load, they are less effective than whole-task approaches in meeting complex learning objectives that require transfer of training to new situations (van Merriënboer, 1997). Whole-task approaches focus on integrating component skills from the outset. Instruction begins with the most simple, but realistic, case that must be learned. Then cases with intermediate complexity are constructed by removing or modifying certain simplifying conditions, and so forth until more complex tasks are presented. In describing this approach, van Merriënboer used the concept of task classes to define simple-to-complex categories of learning tasks. Tasks within a given class were considered equivalent because they could be performed using the same set of generalized knowledge, such as mental models, cognitive strategies or schemata. Learning tasks within a simpler task class induce a lower cognitive load than tasks within a more complex task class because the underlying cognitive schemata do not contain the elements that are needed for the more complex tasks.

Cognitive load can be further reduced by starting a task class with worked-out examples or completion tasks (present a partially-worked example and require the learner to complete the problem), and gradually eliminating these forms of support until a whole problem is given. This approach can be iterated within each progressively more complex task class.

Challenge: To construct realistic training scenarios that do not overwhelm the learner early in the training program.

Developing Highly Interactive, SCORM-Conformant Courseware

The US Department of Defense Advanced Distributed Learning (ADL) initiative specifies levels or categories of interactive courseware that reflect increasing complexity, interactivity, and learner control. Level 1 is a linear presentation—one idea is presented after another with little or no interactivity or learner control of navigation (a "Page Turner"). Level 2 presents more complex ideas, possibly involving simple simulations, with questions or responses required of the learner and feedback providing the correct solution or additional information. In Level 3, emulations and simulations are more integral parts of lessons, and the learner enjoys an increased level of control over the lesson interaction. The focus of more complex lesson scenarios is to affect a transfer of learning. Level 4 emphasizes more in-depth recall of a larger amount of information, increased level of learner control over the lesson, and more extensive simulation.

The ADL introduced the Sharable Content Object Reference Model (SCORM) as a common specification to promote interoperability and reuse of training material across the federal government, and academia, private industry. The SCORM mandate offers great promise for standardization and sharing of course material, but not without some associated costs or limitations for courseware aimed at higher levels of interaction (e.g., Level 3 and higher in which substantial interaction and branching based on learner performance).

More particularly, interactive courseware that employs highly flexible content that may be tailored to different leaner roles, and/or that follows the guided-discovery architecture must address some serious implementation challenges within a SCORM-based environment. The following list outlines some of the issues that we have encountered while designing new guided-discovery courseware or modifying existing, non-SCORM conformant Level 2/Level 3 IMI to meet SCORM guidelines[2]:

- SCORM 1.2 requires sequential navigation within lessons and does not support hyperlinks between lessons that are implemented as “sharable content objects” or SCOs. This restricted navigation makes it difficult for a learner to branch to related training material or lessons, and it similarly limits the design of guided discovery learning applications.
- Limited or no ability to dynamically alter the content, at any level (course, module, lesson, page) based on the learner’s profile, how the learner navigates the site, and the learner’s training history or performance. For example, the application described by Greitzer et al. (2003) uses two levels of training depending on the learner’s job definition (information acquired at registration). Content tailored to one or the other level of training can be shown or hidden “on the fly” as appropriate for the learner. To provide this type of course variation within SCORM 1.2, two separate (but highly duplicative) courses would need to be developed and maintained. For guided discovery learning, dynamic content and navigation are critical features that would have to be supported within the training application.
- SCORM 1.2 limits the ability to develop SCOs that allow dynamic content generation based on the learner's performance. The incorporation of learner support or scaffolding, as described earlier, depends to a large degree on assessing the learner’s performance and adapting the content appropriately.
- An associated challenge (with or without SCORM conformance) is to pass performance data from within an interaction element (e.g., a Flash or Shockwave interaction) back to the Learning Management System (LMS) so navigation and/or content decisions can be made that reflect the learner’s level of learning at the time.

To summarize, we have found that some functionality of highly interactive multimedia training applications might be more limited in a SCORM implementation.

[2] This list is not exhaustive, and is best characterized as representative of the types of problems that may be associated with SCORM-implementation of IMI applications such as described in Greitzer et al. (2003). Some of these issues may be addressed in SCORM 2004.

Challenge: To develop courseware that provides Level 3/higher interactions (including guided discovery learning) that conforms to SCORM standards.

The ability to adapt or change training content based on the learner’s performance is particularly challenging. This is nevertheless a critical need for more advanced training applications that allow the learner some control over navigation or that make dynamic, performance-based decisions about what content to present or make available to the learner.

Thus, a critical requirement for an LMS to support highly interactive performance-based or guided-discovery learning is the ability to receive feedback on the learner’s performance within a multimedia interaction. Responses to cues presented within a simulation or interactive 3-D model, for example, must be known by the training application to generate the next training interaction. Solving this problem remains a challenging area of research. While custom applications have been developed and installed on the server side of the LMS to provide this capability, these have not easily integrated with existing LMS implementations. Also, attempts have been made to perform dynamic content generation on the client. This solution is limited by many of the problems that existed before Web browsers matured: writing in Java (ECMAScript), writing a Java Applet, programming in ActionScript and using FLASH, or writing a custom (non-browser) application. While all of these technologies have well defined purposes, they do not lend themselves to highly interactive courseware using server-side processing.

Challenge: To develop methods or make it easier for the LMS or learning system to receive feedback on the learner’s performance within a multimedia interaction.

We do not have solutions to all of these problems; perhaps some of the SCORM challenges will be dealt with as SCORM technologies and tools become more mature; we hope that the issues described above will foster further discussion and encourage research directions that will help develop technical solutions. We are currently investigating possible solutions to the problem of supporting data transfer into/out of multimedia interaction elements. One avenue of this research has to do with methods of constructing scenarios; another is to investigate development of a middleware application—to be deployed with the learning content—that can communicate with the server-based LMS.

Design/Construction of Scenarios

Design/construction of interactive scenario-based learning material for interactive multimedia instruction is difficult and labor intensive. In the IMI application described by Greitzer et al. (2003), several members of the development team worked for several months to define the set of entities and their relationships, which comprised the scenarios, and to storyboard the scenarios in sufficient detail to allow them to be implemented within the IMI—implementation included construction of interactive learning objects such as 3-D renderings and models that could be manipulated to achieve some desired state. Implementation of the scenarios within integrated exercise(s) was a time-consuming and largely manual process.

Merrill (2002b) described an instructional design process for constructing material within a guided-discovery framework. The approach is referred to as "pebble-in-the-pond" to distinguish it from the traditional "ADDIE" ISD model (Analysis, Design, Development, Implementation, Evaluation); it uses the analogy of expanding circles emanating from the point where a pebble strikes a calm pool of water to represent the progression of the instructional design (and development) from the problem/task definition to the analysis of learning components, design of instructional strategy, and design/implementation of the instructional material (see Figure 1). This model prescribes a systematic method for identifying and defining problems to be implemented within the scenarios, information and resources that the learner needs to complete a task, identifying learning components that apply to the problem, and structuring the problem using part- or whole-task components with varying amount of guidance as learning progresses. It is therefore much more useful and relevant to the process of developing scenario-based, guided discovery learning courseware.

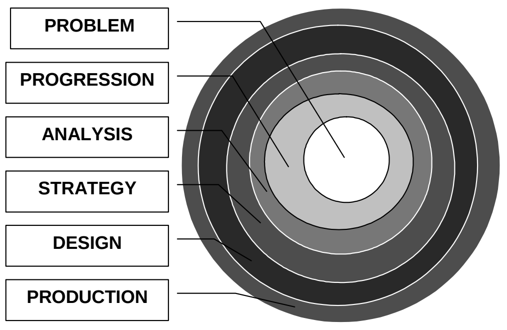

Figure 1. Pebble-in-the-Pond Instructional development model.

While Merrill's (2002b) model provides a methodological foundation for guided-discovery ISD that helps to organize the design/development effort, it does not offer automated tools to facilitate the process. Towards this end, an internal research project conducted at PNNL by Frank Greitzer and colleagues sought to define some support tools to help automate the process.

Challenge: Developing automated support for the implementation of guided-discovery or scenario-based learning content.

The next section describes some ideas about how to represent learning material to facilitate the construction and delivery of scenario-based training.

APPROACH

In this section we describe ideas for tools that help to structure the ISD process to support development of interactive learning content, including a guided-discovery approach described above. First, we describe a scheme for representing the problem space, which forms the basis for designing and constructing learning scenarios that meet Level 3 IMI objectives. Next, we describe some concepts for advanced ISD tools to help the course designer develop scenario-based training content. Finally, we discuss some implications for future ADL/Level 3 IMI development.

Defining the Problem Space

The creation of (ideas for) scenarios is, essentially, a knowledge acquisition process for an instructional designer in which the knowledge must be solicited or extracted from a subject-matter expert. It may entail developing a representation (model) of the problem space that contains all of the concepts (facts and relationships) that are to be learned. A representation or model of the problem space can be considered as a schema that shows the entities (objects), their attributes, and the relations among them.

To aid in describing the representation concepts, we will use a simple example involving a television (TV), a videocassette recorder (VCR) and a cable TV box. To operate this set of equipment, a number of settings and connections must be made correctly. For example, individual components must be plugged into power and proper connections must be made between components. Dependencies between components may need to be met: for example, if the VCR is set to output its signal on channel 3, the TV must be tuned to channel 3. We can describe these entities and relationships in terms of *systems*, which may in turn comprise *components* or other systems. Each system or component may be defined by a set of *state* values. For example, the TV is plugged into power (or not) or it is tuned to channel 3, 4, etc. A defined set of states correspond to "perfect" or normal operations and a different set of states corresponds to "abnormal" or faulted conditions.

Schema and Data Model

Figure 2 illustrates a simplified view of a problem schema for the TV-VCR example. Because of space limits and to keep the diagram simpler, we have not shown the cable box nor have we broken down entities into systems and systems into components (more complicated cases such as the e-Learning application reported in Greitzer et al., 2003 would require this additional detail).

In the illustration, the U represents the knowledge space for the application and the W represents the "world" of cases that we might imagine for this problem. For example, in one situation (e.g., W_1) we might focus only on the TV and VCR; in another, we might only have a TV and a cable box, while in a third we might have all three systems. This allows for the construction of learning tasks varying in complexity or difficulty, which are used to implement part-task or simpler whole-task exercises, as described earlier in this paper. We represent the systems by the objects labeled O_i (O_1 and O_2 in the figure). Here, O_1 is the TV and O_2 is the VCR. Each object has a number of attributes that have associated states. The TV must be plugged in, it is tuned to one of a number of channels, and it has input/output jacks and connectors. These specifications may be denoted by X_{ijk}, where i = the object, j = the attribute and k = the state value. For simplicity we can say the $k = 0$ for the normal state and $k = 1$ for the abnormal state. For example, for a TV (object 1) that is connected to power (attribute 1, state 0), we might use the notation X_{110}. For a TV that is

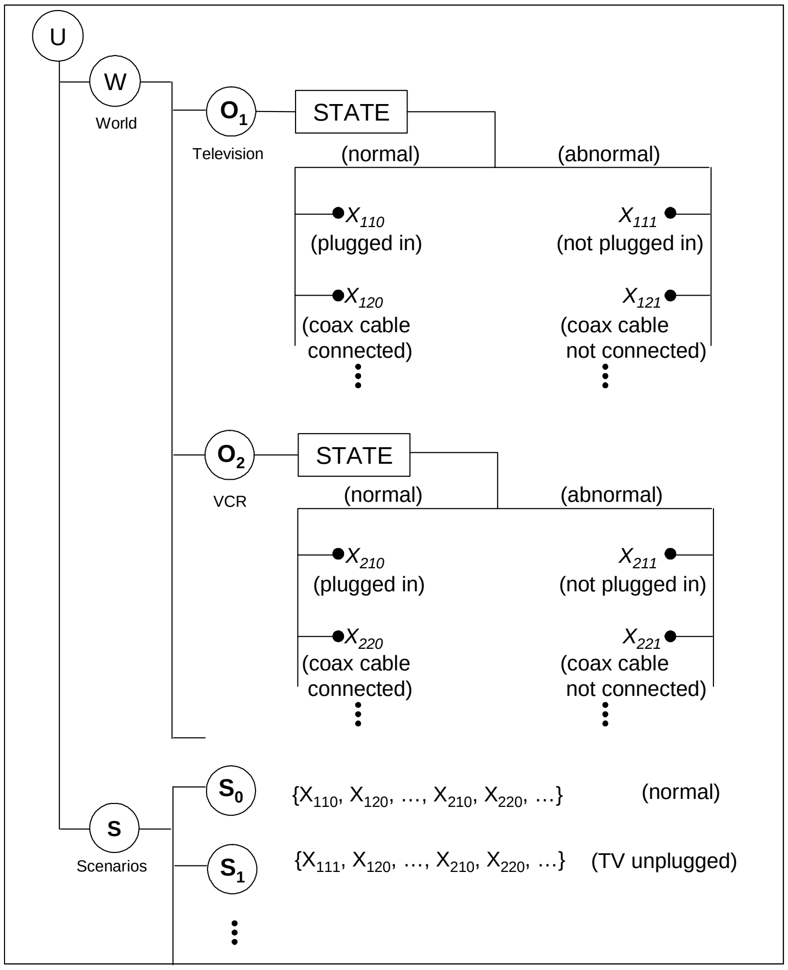
U
W
World
O1
Television
STATE
(normal)
(abnormal)
X110 (plugged in)
X111 (not plugged in)
X120 (coax cable connected)
X121 (coax cable not connected)
O2
VCR
STATE
(normal)
(abnormal)
X210 (plugged in)
X211 (not plugged in)
X220 (coax cable connected)
X221 (coax cable not connected)
S
Scenarios
S0 {X110, X120, …, X210, X220, …} (normal)
S1 {X111, X120, …, X210, X220, …} (TV unplugged)

Figure 2. Simplified diagram of a knowledge-space schema for a hypothetical problem.

not connected to power, we might use the notation X_{111}. Similarly, X_{211} represents a VCR that is unplugged. Using this notation, we can define unique identifiers for every state associated with every object.

This schema provides a mechanism for the course developer to describe the connections and cause/effect relationship between objects in a scenario. This is done by describing the normal and abnormal states of all objects, then chaining the states together (including some abnormal states). For example, the scenario $S_1 = \{X_{111}, X_{120}, \ldots, X_{210}, X_{220}, \ldots\}$ specifies that the TV is not plugged in and the VCR is plugged in. An advantage of this notation is that it supports the definition of a function that "measures" the distance of a particular scenario from the "perfect" state.

If only the TV is unplugged, the scenario is relatively close to the perfect state (there is only one abnormal condition) and the problem is relatively easy. If there are multiple abnormal conditions, the scenario is more difficult because the learner must recognize multiple problems. This approach provides a basis for cognitive load balancing and management of information. It also supports more adaptive responses to the learner's performance. If the learner is able to return all the objects back to their normal states, we can conclude that learning has occurred. If the learner cannot return the objects back to their normal states, the schema allows for the courseware to adapt and, for example, guide the learner to review a particular object's normal state to reinforce learning on a finer grain scale.

Learning Scenarios

To keep the diagram simple, Figure 2 does not include other details that are needed to implement the problem space and define scenarios within this schematic framework. Such details include a text description of the state of the component (e.g., "TV is not plugged into power") and filename identifiers (if applicable) for media files that represent this condition (e.g., an illustration, a photograph, a 3-D model, etc.). The schematic representation in Figure 2 can be implemented as a data model that contains all of these details—the objects in the problem space, their interrelationships, states, text descriptions, and filenames of multimedia representations of the objects.

Figure 3 illustrates a portion of the data model for the

```xml
<system>
        <name>VCR</name>
        <description>JVC VHS DualScan</description>
        <component>
                <name>Power Connection</name>
                <description>The connection between the VCR and the receptacle on the wall.</description>
                <element type="simple">
                        <state condition="normal">
                                <name>Power OK</name>
                                <description>The VCR can be turned ON and OFF.</description>
                                <file>VcrPowerOk.jpg</file>
                        </state>
                </element>
        </component>
        <component>
                <name>Input 1</name>
                <description>Coax cable connection between Cable Box and VCR</description>
                <element type="simple">
                        <state condition="normal">
                                <name>CB-VCR Connection</name>
                                <description>Coax cable connect correctly to VCR</description>
                                <file>VcrInput1CoaxOk.jpg</file>
                        </state>
                </element>
        </component>
        <component>
                <name>Output 1</name>
                <description>Coax cable connection between TV and VCR</description>
                <element type="simple">
                        <state condition="abnormal">
                                <name>TV-VCR Connection</name>
                                <description>Coax cable disconnected from VCR</description>
                                <file>VcrOutput1CoaxNotOK.jpg</file>
                        </state>
                </element>
        </component>
</system>
```

Figure 3. Excerpt from XML representation of the TV-VCR knowledge schema.

TV-VCR example, implemented in XML. In this form, the data model provides a powerful schema for representing expert knowledge or conceptual models of simple or complex systems. The XML data model is flexible and can represent logical and real world objects as well as scenarios involving interactions and cause/effect relationships between those objects. The advantage of the data model is that it can represent these objects using text descriptions, graphic images (gif, jpg, etc), and/or interactive media (Shockwave, Flash, etc). This allows course developers to define the problem space using simple text. The text descriptions are helpful in defining images that graphic artists can create because the descriptions provide a context of how the graphic image will be used. The course could be entirely text based, or it could be extended to include graphic images that complement and embellish the text descriptions and support interactive exercises and scenarios.

Automated Support for Instructional Design

We envision a set of automated instructional design tools to aid the processes of defining scenarios and in generating/delivering the scenario-based training. The concept comprises a set of ISD tools as illustrated in Figure 4.

The first component, referred to as the Problem Space & Scenario Definition Wizard, helps the instructional designer to define the overall problem space, its detailed composition and interrelationships among its components. This tool also serves to define scenarios by assisting the subject-matter expert in specifying the learning objects within a scenario that help to exercise the learner's knowledge and skills with the training content and to support performance assessment.

The second component includes a data model that supports the scenario-based training content. The data model represents the concepts/facts and inter-relationships within the problem space to be learned; it contains links to multimedia files that support learning through visualizations, demonstrations, simulations, and interactive exercises (building blocks for scenario-based instruction). The state of an object in the data model can be represented by simple features (e.g., ON, OFF) or more complex structures. An example of a more complex state is the "difficulty" of a concept, which, when coded, can be used as a criterion for selecting scenarios to tailor training experiences to the learner's performance. As another example, the data model can be extended to accommodate training material and media that support Section 508 of the Rehabilitation Act of 1973.

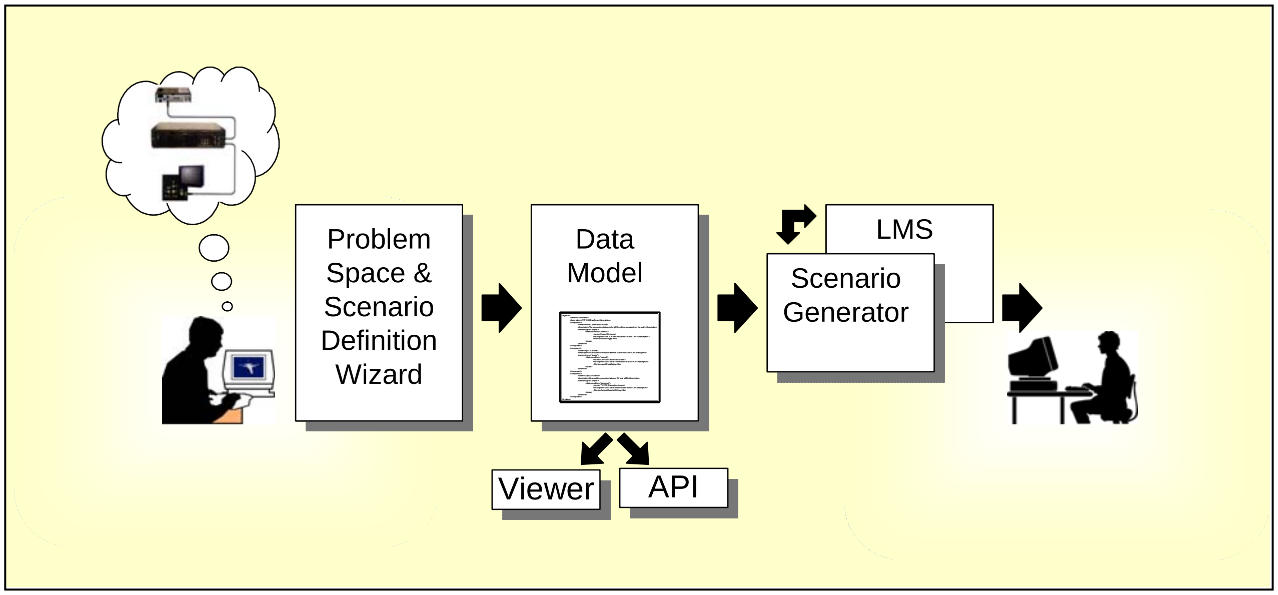

Figure 4. Illustration of ISD support components. Designer uses Problem Space & Scenario Definition Wizard to specify the problem space and objects/concepts that are needed to develop learning scenarios. Data model is generated (XML representation) and an abstract object Viewer supports more detailed specification of objects, attributes, and requirements for their graphical/multimedia representations. An application programming interface (API) supports access to the data model by tools such as the scenario generator or job aids. The scenario generator creates scenarios as defined in the data model so that they may be deployed by the LMS.

The XML representation of the data model can be used to facilitate the automatic generation of interactive training scenarios. The scenario generator (third major component of the automated support system) forms interactive scenarios by stringing together representations in the data model. The idea is that scenarios may be "assembled" from basic learning objects in the data model. For example, consider two scenarios in the TV-VCR example: scenario S_0 (described earlier) with all components connected and working properly, and scenario S_2 in which all connections are OK except that the VCR is set to send its output to channel 4, while the TV is tuned to channel 3. It can be seen that object attributes in scenario S_0 and S_2 are nearly all identical, except for one important difference. Similarly, most of the multimedia or graphics files needed to represent the TV and VCR objects in both scenarios are identical; the exception is the file that shows the detail or close-up of the VCR control that selects "3" or "4" as the output channel. For the purposes of defining scenarios, it would be desirable (for efficiency and storage purposes) to take advantage of this commonality. Typically, however, we do not exploit this fact because we implement complete scenarios independently (this was done in the interactive exercise scenarios developed by Greitzer et al. (2003). The goal for implementing the scenario generator is to enable smaller sets of multimedia files (rather than entire scenarios) to be brought together as needed to produce a given scenario. Key to achieving this more dynamic scenario generation capability is the ability to pass performance data from within an interaction element to the LMS or delivery system to properly effect navigation within a scenario.

Other support concepts include an Application Programming Interface (API) to the data model, which would be useful for developers to write applications that can take advantage of the data model; and an abstract object Viewer to support scenario definition that would enable the courseware development team (particularly the graphics/multimedia developers) to correlate the text descriptions and a visual representation of the objects and better understand more complex relationships.

CONCLUSIONS

Developing interactive, scenario-based instruction is an extremely labor intensive process because of the great effort involved in defining the elements and their associated characteristics and inter-relationships that make up the scenarios. The representation schema that we have described helps to structure this process, if not to automate it. This structured approach is being used to design and develop a Web-based training application that employs scenario-based guided-discovery learning [see the paper by Greitzer, Pond and Jannotta (2004), elsewhere in these proceedings]. Further work is necessary to develop more automated tools for instructional design that use this structured process to facilitate knowledge elicitation and implementation of guided-discovery scenarios for learning.

The knowledge elicitation, representation, and scenario-based learning components of the approach described will have applicability beyond courseware itself, because the relationships defined in the data model also can be used to form the foundation of job aides or expert systems that workers can consult in the field. Thus, the broader context is to produce the foundation for a cognitive-based performance support system for training and expert system development for learning and sustaining complex cognitive skills.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the valuable suggestions, technical insights and implementation support from Chance Younkin at PNNL.

REFERENCES

Clark, R. (1998). *Building Expertise: Cognitive Methods for Training and Performance Development.* Washington D.C.: International Society for Performance Improvement.

Greitzer, F.L. (2002). A cognitive approach to student-centered e-Learning. *Proceedings, Human Factors and Ergonomics Society 46th Annual Meeting*. 2064-2068.

Greitzer, Fl. L., Pond, D. J., and Jannotta, M. (2004). Scenario-based training on human errors contributing to security incidents. *Proceedings, Interservice/Industry Training, Simulation, and Education Conference*. December 2004.

Greitzer, F.L., Rice, D.M. Eaton, S.L., Perkins, M.C. Scott, R.T., Burnette, J.R., and Robertson, S. R. (2003) A cognitive approach to e-Learning. *Proceedings, Interservice/Industry Training, Simulation, and Education Conference*. December 2003.

Jonassen, D. (1999). Designing constructivist learning environments. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II) (pp. 215-239). Mahwah, NJ: Lawrence Erlbaum Associates.

Merrill, M. D. (2001) Components of Instruction: Toward a theoretical tool for instructional design. *Instructional Science*. *29(4),* 291-310.

Merrill, M. D. (2002) (a). First principles of instruction. *Educational Technology Research and Development*. 50(3), 43-59.

Merrill, M. D. (2002) (b). A pebble-in-the-pond model for instructional design. *Performance Improvement*. *41(7),* 39-44.

Nelson, L. M. (1999). Collaborative problem solving. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II) (pp. 241-267). Mahwah, NJ: Lawrence Erlbaum Associates.

Reigeluth, C. M. (1999). (Ed.) *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ: Lawrence Erlbaum Associates.

Schank, R. C., Berman, T. R., & MacPerson, K.A. (1999) Learning by doing. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II) (pp. 161-181). Mahwah, NJ: Lawrence Erlbaum Associates.

Van Merriënboer, J. J. G. (1997). *Training Complex Cognitive Skills*. Englewood Cliffs: Educational Technology Publications.

Van Merriënboer, J. J. G., Kirschner, P. A., & Kester, L. (2003). Taking the load off a learner's mind: Instructional design for complex learning. *Educational Psychologist*, *38*, 5-13.