INSTRUCTIONAL TRANSACTION THEORY (ITT): INSTRUCTIONAL DESIGN BASED ON KNOWLEDGE OBJECTS

THE INSTRUCTIONAL SIMULATOR OR IDVISUALIZER

M. DAVID MERRILL

INTRODUCTION

Component Display Theory (CDT, see Merrill, 1983, 1987; Merrill, 1994) provides a list of prescriptions for designing instruction for different kinds of instructional outcomes. As we moved toward trying to automate the instructional-design process, we found that CDT was not precise enough to allow computer implementation of expert system technology that would prescribe instruction. Instructional Transaction Theory (ITT) is an attempt to provide more precision to CDT thereby making automated instructional design a possibility. This increased precision also has value for instructional designers in that it provides a more precise way to describe knowledge representation, instructional strategies, and instructional-design prescriptions.

CDT was an attempt to identify the components from which instructional strategies could be constructed. CDT describes instructional strategy in terms of strategy components: *primary presentation forms (*PPFs*), secondary presentation forms* (SFPs), and *interdisplay relationships* (IDRs). CDT identifies strategy prescriptions for different kinds of learning outcomes. Each of these prescriptions identifies a best case combination of PPFs, SPFs, and IDRs for a particular kind of learning outcome. CDT was analysis oriented, emphasizing the components of instructional strategies for different kinds of instructional goals.

ITT is synthesis oriented, emphasizing the integration of these components into instructional transactions. An instructional transaction is all of the interactions necessary for a student to acquire a particular kind of knowledge or skill.

The presentation of ITT in this chapter emphasizes what Reigeluth calls *component methods.* In this paper we will introduce a methodology for representing knowledge in the form of knowledge objects and elements (slots) of knowledge objects. Knowledge objects and their elements provide the components of subject-matter content (knowledge). ITT describes instructional strategy as methods (algorithms) for manipulating the elements of knowledge objects.

INSTRUCTIONAL THEORY

Instructional theory is concerned with two primary considerations: What to teach and how to teach.

What to teach has two considerations: selection and representation. ITT is not concerned with the curriculum selection question of what should be taught, but rather, having selected what should be taught, what are the knowledge components required for a given type of instruction? And how should these knowledge components be represented to facilitate instructional design?

How to teach specifies the way that these knowledge components are presented to the student in order to engage the student in an interaction which is appropriate for promoting the acquisition of the knowledge or skill that is the goal of the instruction. Instructional strategies include the presentation of the appropriate knowledge components, practice with or student activities involving these knowledge components, and learner guidance to facilitate the student's appropriate interaction with these knowledge components.

The Computer Program Assumption

Authoring systems for computer-based instruction are based on a database model of computing. The student is presented with a record containing subject-matter content (a program frame). The program then presents the student with one of several options: press a key to see the next record; select an item from a menu to see the next record; or respond to a question and the next record will be determined based on the answer given. This model of instructional computing has one serious limitation: except for the branching strategy, all other instructional strategies are hidden in the record (frame) and therefore transparent (not available) to the instructional system for additional processing. The instructional strategies

to be used must be determined by the designer of the system and incorporated within the records of the database.

Outside of instruction, many computer programs are based on an algorithmic model. In this model data are manipulated by one or more sets of instructions for processing (displaying, transforming) these data. If the knowledge to be taught is thought to be data, and the strategies for teaching this knowledge are thought to be instructional algorithms, then an algorithmic model of computing can also be applied to instruction. However, an algorithmic instructional system requires that the knowledge be accessible in a form that lends itself to processing by the instructional algorithms. A primary focus of ITT is to describe such a knowledge representation system.

ITT is an algorithmic instructional system. Knowledge is represented as data. The components of this knowledge are processed (displayed, transformed) by the instructional algorithms built into an instructional system. While this representation was specifically designed to facilitate the design of computer-based instruction, this form of representation also has value for designing instruction for other modes of delivery.

The Gagné Assumption

Gagné (1965, 1985) stated as a primary assumption of instructional theory that there are different kinds of learning outcomes (learning goals) and that each of these different kinds of learning outcomes requires unique conditions for learning. An appropriate instructional strategy incorporates all of the necessary conditions for presenting the knowledge or demonstrating the skill, providing practice with feedback, and providing learner guidance for a given type of learning outcome. Gagné indicated nine events of instruction which include these three phases of instruction. Appropriate conditions for learning always require all of these activities. Information that does not include presentation, practice, and learner guidance is information but not instruction. Different instructional outcomes (objectives) require different types of presentation, different types of practice, and different kinds of learner guidance. It is this difference in the required conditions for learning that distinguishes different kinds of learning outcomes.

In previous papers we have called an instructional strategy that incorporates all of the conditions for teaching a given type of learning an instructional transaction. We previously

identified 13 classes of instructional transactions (Merrill, Jones, & Li, 1992). In this paper we shall describe only three of these classes in terms of instructional algorithms for processing the elements of knowledge objects. These instructional transactions include: IDENTIFY[1] (component or naming or parts-of …, also related to "facts" in CDT), EXECUTE (activity or procedures or how-to …, "procedures" in CDT), and INTERPRET (process or what-happens …, "principles" in CDT).

INSTRUCTIONAL TRANSACTIONS

In ITT an instructional transaction is all of the learning interactions necessary for a student to acquire a particular kind of knowledge or skill (learning goal). The instructional algorithm (called an instructional transaction shell) required to promote an appropriate instructional transaction operates on a set of knowledge objects that are related in a particular way (knowledge structure) and that contain all of the knowledge that is required for a student to acquire the instructional goal. An instructional transaction algorithm includes the presentation strategies, the practice strategies, and the learner guidance strategies that are required and appropriate to promote acquisition of the instructional goal.

In ITT, instructional strategies represent various ways to show, or request the student to provide, the elements of knowledge objects. Hence, an instructional strategy is an algorithm for processing the knowledge data (elements) of knowledge objects.

Knowledge Objects

Knowledge objects are containers consisting of compartments (slots) for different related elements of knowledge. The framework of a knowledge object is the same for a wide variety of different topics within a subject domain, or for different subject domains. The contents of a given compartment differ, but the nature of the knowledge element in a given compartment is the same.

All knowledge objects have a set of information slots including: name, portrayal, and description. The name contains one or more symbols or terms that reference the knowledge. The portrayal is one or more multimedia objects (text, audio, video, graphic,

---

[1] We used the terms IDENTIFY, EXECUTE, and INTERPRET in order to relate these transactions to our previous paper (Merrill, Jones, & Li, 1992).

animation) that will show or represent the knowledge object to the student. The description slot is an open compartment into which an author can place any desired information about the knowledge object. It is possible for the description slot to be subdivided into several subslots. These might include function, purpose, etc., and may be defined by a given user.

We have identified four types of knowledge objects: entities, properties, activities, and processes,[3]. (See Jones, Li, & Merrill, 1990.) Entities represent objects in the world and can include devices, persons, creatures, places, symbols, etc. Properties represent quantitative or qualitative attributes of entities. Activities represent actions that the learner can take to act on objects in the world. Processes represent events that occur in the world that change the values of properties of an entity. Processes are triggered by activities or by other processes.

A knowledge object may also have links to other knowledge objects. The nature of these links will be described in more detail in later sections of this paper.

Goals of ITT

Effective instruction. First, we are concerned with the current emphasis on information and the lack of emphasis on appropriate instructional strategies. By describing instructional strategies as algorithms (transactions) for manipulating data structures (knowledge objects), we have provided a much more precise description of the different kinds of instructional transactions required for different kinds of instructional outcomes (goals or objectives). Our hope is that this formulation will enable instructional designers to design more effective and appealing instructional products. Furthermore, by building these transactions into instructional development tools, there is an increased probability that the resulting instructional interactions will be based on sound principles of instructional-design.

Efficient instructional development. Second, our intent was to derive theory and methodology which would facilitate the automation of much of the instructional-design process. Instructional development is a labor-intensive industry. If we are to obtain efficiencies in the development of large amounts of computer-based, interactive, multimedia instruction, then we must significantly increase our design and development efficiency. We believe that building appropriate instructional transactions into instructional

development tools will enable automating portions of the instructional-design process and will enable us to realize this efficiency.

Instructional learning environments. Third, the development of interactive learning environments (instructional simulations and microworlds) is extremely labor intensive and hence very expensive to develop using existing technologies. Representing knowledge as knowledge objects enables the building of a general-purpose simulation engine. This makes possible a learning environment builder that enables the efficient development of these more effective instructional interactions. Furthermore, since tutorial instruction and learning environments are based on the same knowledge representation, this architecture enables effective learner guidance to overlay instructional learning environments.

Adaptive instruction. Fourth, ambiguous representation of knowledge and imprecise specification of instructional strategies has hindered the development of truly adaptive instruction. The precise representation of knowledge in the form of knowledge objects and the representation of instructional transactions as algorithms for manipulating this knowledge makes possible instructional strategies that can be adapted to individual learners in real time as they interact with the instructional materials.

Scope of ITT

The component methods of ITT can be used to describe any instructional strategy, whether tutorial or experiential. The component methods of ITT have been used to implement technical training as well as *soft skill* training. These methods lend themselves easily to instruction in *well- structured* domains however these same methods can be applied to *ill-structured* domains. A detailed explanation of how these methods are applied to soft skills and ill-structured domains is beyond the scope of this chapter.

We previously identified 13 classes of instructional transactions (Merrill, Jones, & Li, 1992). These instructional transactions are listed in Table 1. ITT has been used to implement the component transactions*: IDENTIFY, EXECUTE, and INTERPRET. An instance of this implementation is described in this paper. We have designed implementations for some of the abstraction transactions but only demonstration systems have been implemented. We believe we can also implement the association transactions with knowledge objects and ITT methods.

The Instructional Simulator was used to develop an anthropology study of an African village ("Life in a Mende Village"). This learning environment enabled the student to go to different locations in the village and to converse with villagers in order to gather information about their agriculture and lifestyle. At the conclusion of their study the student submitted a report to the Humanitarian Council. The simulation engine, PEAnet architecture, and learner guidance described later in this paper for the "Valve Simulation" were all used in this simulation of an ill-structured domain.

The transactions for IDENTIFY, EXECUTE, and INTERPRET are building blocks for abstraction and association transactions. All instruction involves acquisition of the knowledge and skill promoted by these fundamental transactions. These transactions account for the instructional strategies found in most of the existing instruction in training.

Table 1 Thirteen Classes Of Instructional Transactions

Component Transactions	
IDENTIFY	name and remember information about parts of an entity
EXECUTE	remember and do steps in an activity
INTERPRET	remember events and predict causes in a process

Abstraction Transactions	
JUDGE	order instances
CLASSIFY	sort instances
GENERALIZE	group instances
DECIDE	select among alternatives
TRANSFER	apply steps or events to a new situation

Association Transactions	
PROPAGATE	acquire one set of skills in the context of another set of skills
ANALOGIZE	acquire steps of an activity by likening to a different activity
SUBSTITUTE	extend one activity to learn another activity
DESIGN	invent a new activity
DISCOVER	discover a new process

REFERENCE EXAMPLE[2]

In this presentation we will emphasize instructional goals for IDENTIFY, EXECUTE, and INTERPRET transactions. The example presented for illustration has as a primary goal learning a procedure. In support of this goal the learning environment includes guidance for learning the function and location of the parts of a device. Also in support of the procedural goal the learning environment includes guidance for learning to predict consequences and trouble-shooting unexpected consequences by identifying the underlying conditions which must be satisfied before a given consequence can occur.

Figure 1 illustrates a learning environment designed to teach the learner how to install or remove a double seat valve. This instruction was prepared to train technicians who must maintain valves in dairies and breweries.

Prior to entering the learning environment, the learner is given the following goal:

"This work task involves properly disconnecting all hoses and wires from the valve and correctly unbolting and removing the valve insert from the pipeline."

The learning environment consists of a diagram of the valve showing all of the hoses and wires connected to the valve. The learning environment is an "open-ended learning environment"[3]. Clicking on a given part of the diagram brings up an action menu. For example, clicking on the switch brings up a menu with the action *flip*. Selecting this action causes the switch to change positions (down to up) and produces an audible click. This action also has the consequence of setting a property of the compressor to *off*. Clicking on the connector for the air hose brings up the action *undo,* which when selected disconnects the air hose from the valve. Hoses and wires can be connected and disconnected at will except as constrained by the conditions of the system. An example of a constraint is that you cannot disconnect the air hose until the compressor is off. By exploring the system,

---

2 The learning environment described was developed using the Instructional Simulator™, an instructional development tool developed by Leston Drake, the present author, and other members of the ID_2 Research Group at Utah State University. (See Merrill & ID2 Research Team, 1993; Merrill, 1997).

[3] An open-ended learning environment is one in which the learner can perform any of a number of actions in any order and see their consequence. A constrained learning environment often enables the learner to perform only a single action at any moment in time.

most learners can eventually figure out how to disconnect the valve and remove it from the pipe.
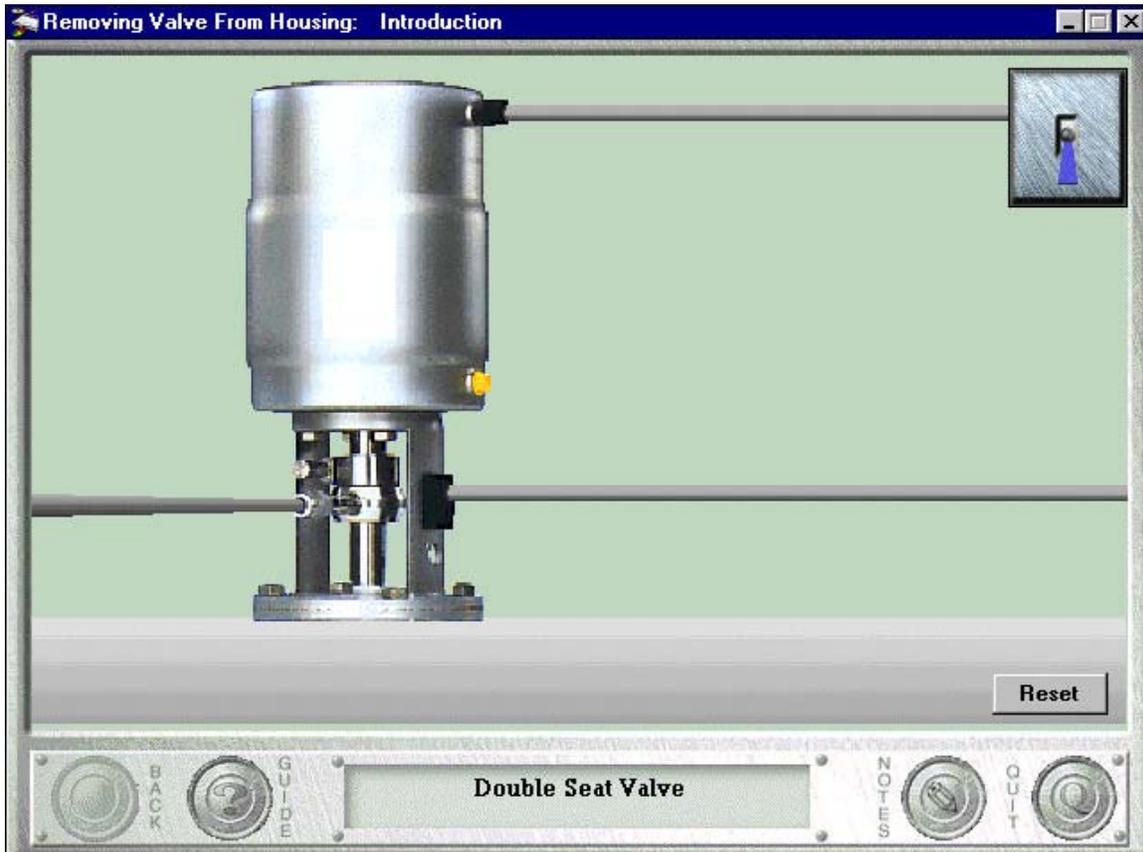

Figure 1 Learning environment for valve removal simulation.

The learning environment is supported by learner guidance of various types. This learner guidance implements various instructional strategies designed to teach the student knowledge about the valve and the skill of removing the valve from the pipe.

Part Location, Function, and Naming

The goal of this learner guidance function is to teach the student the name, location, and function of the various parts of the valve. As the student moves the cursor about the learning environment, the name of the part under the cursor is shown in the window at the bottom of the screen. The window currently shows the name *"double seat valve"*. Thus, by moving the cursor around the screen the student can *explore* the names of the various parts of the valve. Clicking on the right button of the mouse causes a functional description to appear in a pop-up window near the part under the cursor. For example, right clicking on the air connection brings up the illustration and scrolling description illustrated in Figure 2.

Note that the functional description in this system could be any medium including graphic, video, audio, text, or a combination. The authors chose to use a picture and text caption. The student can thus *explore* the function of each part of the device.
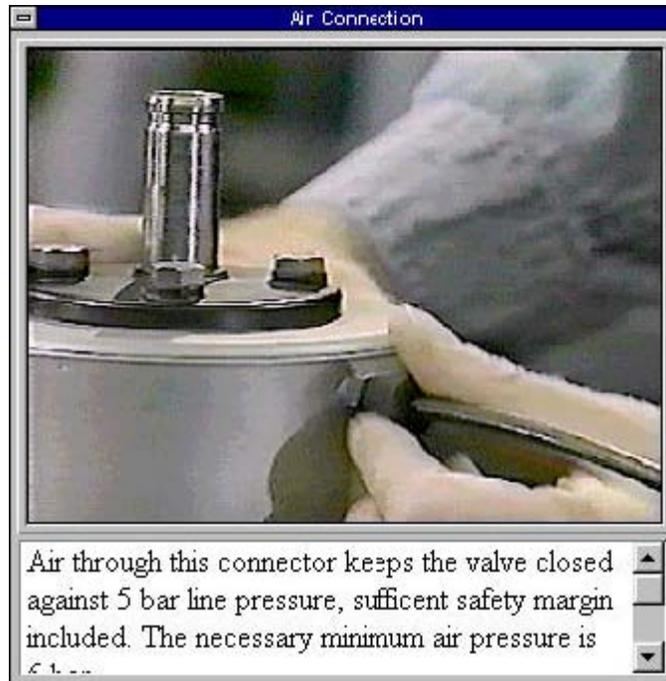

Figure 2 Functional description of a part of the valve.

Clicking on the *Guide* button on the control panel at the bottom of the screen pops up a menu of guidance options. One option is, *Tell me about some of the parts.* Selecting this guidance causes a lecture to be provided about the parts of the system. The guide presents the same type of information as shown in Figure 2 for each part in turn. The student has control over the pace of the lecture by indicating when they are ready for the next part to be presented.

Another guide option is, *Let me locate the parts.* The guide then presents a name of a part and the student is required to point to (click when the cursor is over) the part. Another guide option is*, Let me name the parts*. The guide highlights a part and the learner is required to select the correct name from a list. Another guide option is, *Let me identify the functions*. The guide presents a description of a part and the student must click on the corresponding part. Right/wrong/correct-answer feedback is provided after each response. These practice activities incorporate sampling with replacement such that when a student misses an item, it is put back into the list and presented to the student again until the

student has correctly responded to each part. At the end of a practice exercise the student is given a score indicating how many tries were required to correctly name, locate, or identify the function of each part.

One of the goals of ITT is to enable an instructional development system to automatically generate appropriate instruction. In the Instructional Simulator the instructional strategies (presentation and practice) are built into the system. The designer merely describes the knowledge objects, and the system automatically generates these presentation and practice strategies.

Procedure Learning

Given enough time, most students can eventually discover the procedure for removing the valve from the pipe. However, raw discover is inefficient and often results in a trial-and-error approach to the performance in the real world. Procedure learning is more efficient when appropriate learner guidance is provided. The goal of procedure learning in this learning environment is to learn the necessary and sufficient steps for removing and replacing the valve in the pipe.

The guide provides four levels of procedure practice. Level 1 is a *hands-off* demonstration. In this demonstration the guide performs each of the steps in turn. The cursor moves to the appropriate part, the appropriate action is indicated, the guide then performs the appropriate action, and the system illustrates the consequence of this action. Level 2 is a *Simon Says* demonstration/simulation. In this demonstration the guide tells the student the step to perform. For example, "Flip the air compressor switch." If the student attempts to do any other action the guide presents a message such as, "That is not the air compressor switch." Thus the student can only perform the step requested by the guide. After the student has selected the action, the system illustrates the consequence of that action.

Level 3 is a *Do the next step* simulation. In this practice the guide presents the message, "Do the next step." If the student attempts to do different step the guide presents a hint such as "That is not the air compressor switch." After the student has selected the action, the system illustrates the consequence of that action.

Level 4 is *performance* or *You-do-it* simulation. In this practice the student can perform any of the steps in the procedure and see the consequence of this step subject to the constraints of the system (that is, some steps cannot be performed until a prior step has been completed. For example, you cannot remove the valve until you have removed the flange bolts, inserted a bolt in the tap hole to break the seal, and removed the bolt from the tap hole). When the student believes the procedure has been completed he or she clicks a *finished* button. The guide then shows the steps required for the shortest path to the goal and also shows the steps taken by the student. Unnecessary or incorrect steps in the student's path are highlighted in red. The number of steps required by the student to accomplish the goal is recorded in the student record.

ITT knowledge objects enable all of these levels of practice to be built into the system. The designer merely provides the elements of the knowledge objects required by the simulation and the system automatically creates the various levels of practice available in the system. The designer or learner can select some or all of these levels of practice for a given learning environment.

Explanation, Prediction, and Trouble-Shooting

It is one thing to learn the steps in a performance and to be able to carry them out in order to accomplish some goal (such as removing a valve from a pipe). However, when learners have an explanation of each step in the procedure that identifies what happened and why it happened, their ability to retain the procedural skill is enhanced. In addition, knowing what happened and why is necessary to problem solving or trouble-shooting a device or system. *What happened* indicates the consequence of a given action. Often, what happened can be observed by the student in changes in the appearance of the system. In some cases what happened may change a condition of the system which does not show up in the physical appearance of the system. *Why* indicates the conditions which must be met for a given consequence to occur. When a student performs an action and nothing happens or something unexpected happens, then an explanation indicates what conditions were not met or what conditions led to the unexpected consequence. The goal of explanation is to enable the learner to "predict" what will happen under specified conditions, or to "explain"

(identify the conditions which were not met often called trouble-shooting) when a consequence fails to occur or when an unexpected consequence occurs.

The guide provides three levels of explanation. One guidance option is *Explain*. During free exploration of the system the student can request an explanation after any action. The guide presents a description of what just happened and why it happened. For example, the student attempts to remove the valve. Nothing happens. The student requests an explanation. The guide provides the following message: "When you attempt to remove the valve from the pipe nothing happens. This is because the flange bolt is still in the tap hole." The *Explain* function can also be turned on during any of the practice levels. The *Explain* display is updated after each action by the student.

A second guidance option is *Predict*. The guide configures the system and asks the student to select from a list "What happens next?" and "Why?" The student can then confirm his or her prediction by executing the next step(s) in the procedure and observing what happens. The student's accuracy in prediction is recorded by the system.

A third guidance option is *Trouble-Shoot*. The guide configures the system, sometimes introducing a fault. The student is requested to carry out the next step(s) in the procedure and to explain what happened and why. In this situation the student, rather than the guide, provides the explanation. The student selects what happened and why from a list of consequences and conditions.

ITT knowledge objects make it possible for the designer of a learning environment which includes an explanation system to instantiate (provide information for) the elements of the knowledge objects required for the simulation, and the system will automatically generate the various levels of explanation.

INSTRUCTIONAL TRANSACTIONS

In the following paragraphs we first describe a learning environment. A learning environment is described in terms of (a) the instructional goal it is designed to promote; (b) the knowledge structure required by the learning environment; (c) the general simulation engine which operates on this knowledge structure to represent activities and processes that

occur in the world; and (d) the learning activity of exploration by which the student interacts with the learning environment.

In each of the subsequent sections we define the instructional transactions of IDENTIFY, EXECUTE, and INTERPRET. In each of these sections we describe (a) the instructional goal promoted by the instructional transaction; (b) the knowledge structure required by this kind of transaction; (c) the presentation of information to the student; (d) the practice with feedback required by the transaction; and (e) learner guidance that facilitates learning from the transaction.

Learning Environment Goal

The goal of a learning environment is to enable the student to explore some device or setting. The objects in the environment behave in a way similar to their behavior in the real world. Students are able to act on objects in the environment and see the consequences of their actions. An open-ended learning environment allows free exploration within the constraints of the learning environment. So-called simulations that allow only a single action and are constrained as to the path that the student must take are merely interactive demonstrations, not learning environments.

Knowledge Structure

For learning environments a knowledge object for an entity is expanded to include slots that point to one or more property knowledge objects. A property knowledge object, in addition to a name and a description, has a set of possible values. Each of these possible values is associated with a portrayal or indicator; thus a property knowledge object has a portrayal or indicator for each of the values that the property can assume. Thus, when the

value of a property knowledge changes its portrayal also changes.

In the example, a property of the switch is *position* with two values: *on* and *off*. When the value of the switch property, *position*, is *on*, then the portrayal is a graphic of the toggle in the up position. When the value of the switch property, *position*, is *off*, then the portrayal of this value is a graphic of the toggle in the down position.

A *process* is defined as a change in the value of some property of some entity. We say that this change in property value is the *consequence* of the process. Processes are also

conditional, that is, a given process will not execute unless its conditions are met. A *condition* for a process is defined as a value on some property of some entity. A condition for a process is thus a value of a property. If in a given situation the value of the property is the same as the value specified for the condition for the process, the process executes; if the value of the property is not the same as the value specified for the condition for the process, the process does not execute. Finally a process can *trigger* another process. This provides for a chain of events (processes), each one triggered by the previous event (process). When a process is triggered, it evaluates its conditions, if they are true it executes, if the condition is false it does not execute; it then triggers the next process in the chain whether or not it executed its own consequence.

In the example, one process is *disconnect air line*. It is triggered by the activity *undo air line*. The air line (an entity) has a property, *connection,* with two values: *connected* and *disconnected*. The portrayal for the value *connected* is a graphic of the air hose connected to the double seat valve; the portrayal for the value *disconnected* is a graphic of the air hose disconnected from the double seat valve. This process, *disconnect air line*, has two conditions: (a) the value of the property *connection* of the entity *air line* is *connected* and (b) the value of the property *position* of the entity *air compressor switch* is *off*. If the actual value of the air compressor switch is *on*, then the process*, disconnect air line*, will not execute and nothing happens when the student executes the activity, *undo air line*.

The relationship among processes, entities, and activities enables the construction of learning environments from knowledge objects. In ITT this set of interrelationships is called a PEAnet (process, entity, activity network). Figure 3 illustrates these PEAnet relationships. The learner executes some activity on a controller (itself an entity or part of some other entity). This action triggers a process. If the conditions of the process are true, then the process changes the value of a property. When the value of the property changes then the portrayal of this value also changes, thus causing a consequence for the process to be indicated to the learner.
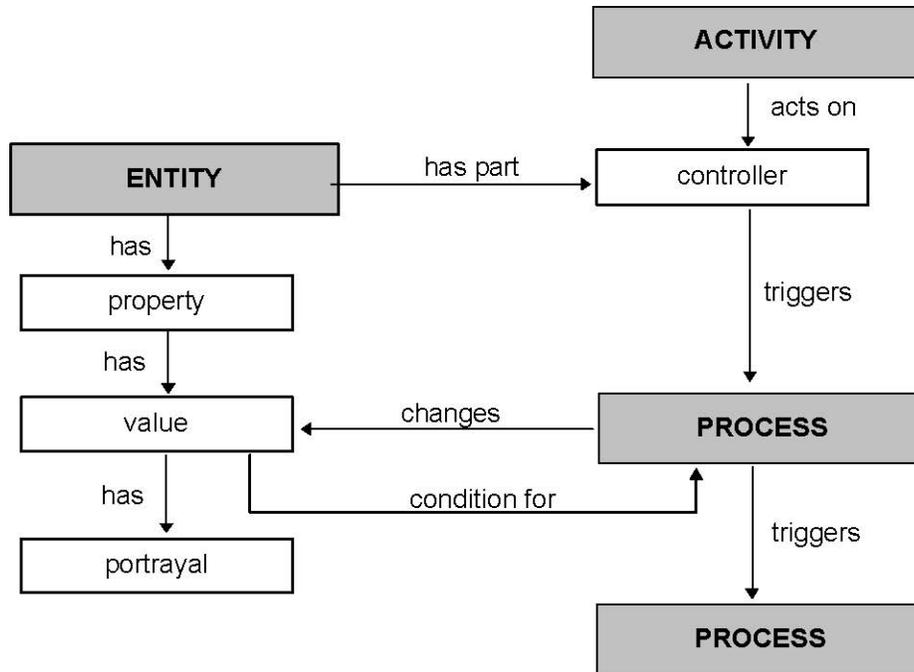
Figure 3 PEAnet relationships among processes, entities, and activities.

Table 2 lists the entities, properties and property values involved in the Valve Removal learning environment.

Table 2 Entity Property Property Values

AC Switch	position	on/off
Air line	connection	connected/disconnected
Cleaning line	connection	connected/disconnected
Valve indicator line	connection	connected/disconnected
Flange	flange bolts	inserted/removed
Tap hole	flange bolt	inserted/removed
Valve	installation seal	inserted/removed seated/unseated

Figure 5 illustrates the PEAnet relationships involved in the "Removing Valve from Housing" learning environment. The sequence of activities required for the student to remove or install the valve from the housing is indicated down the left side of the diagram; the processes triggered by these actions are shown in the center column; and the consequences (change of value of properties) are shown in the third column. The actions include: flip the air compressor switch, undo/connect the air line, undo/connect the cleaning connection, undo/connect the valve indicator line, remove/insert the flange bolts,

insert a flange bolt into the tap hole, remove the flange bolt from the tap hole, pull out/insert the double seat valve.

Some processes have more than one consequence. For example, there are two consequences for the process *toggle.* When the process *toggle* executes it changes the value of *switch position* to *off* if it is *on,* or to *on* if it is *off*. The letters on the right side of the diagram indicate conditions (property values) for a consequence to execute. If the conditions are not met the process does not change the value of the property. For many processes, there are alternative consequences. When the value of some property has one value, one consequence is executed; when the same property has a different value, another consequence is executed. For example, there are two consequences for the process *disconnect air line*. The first consequence, *set connection of air line to the value disconnected*, is executed when the value of the switch position is *off* and the second consequence, *show an accident message*, is executed when the value of the switch position is on.

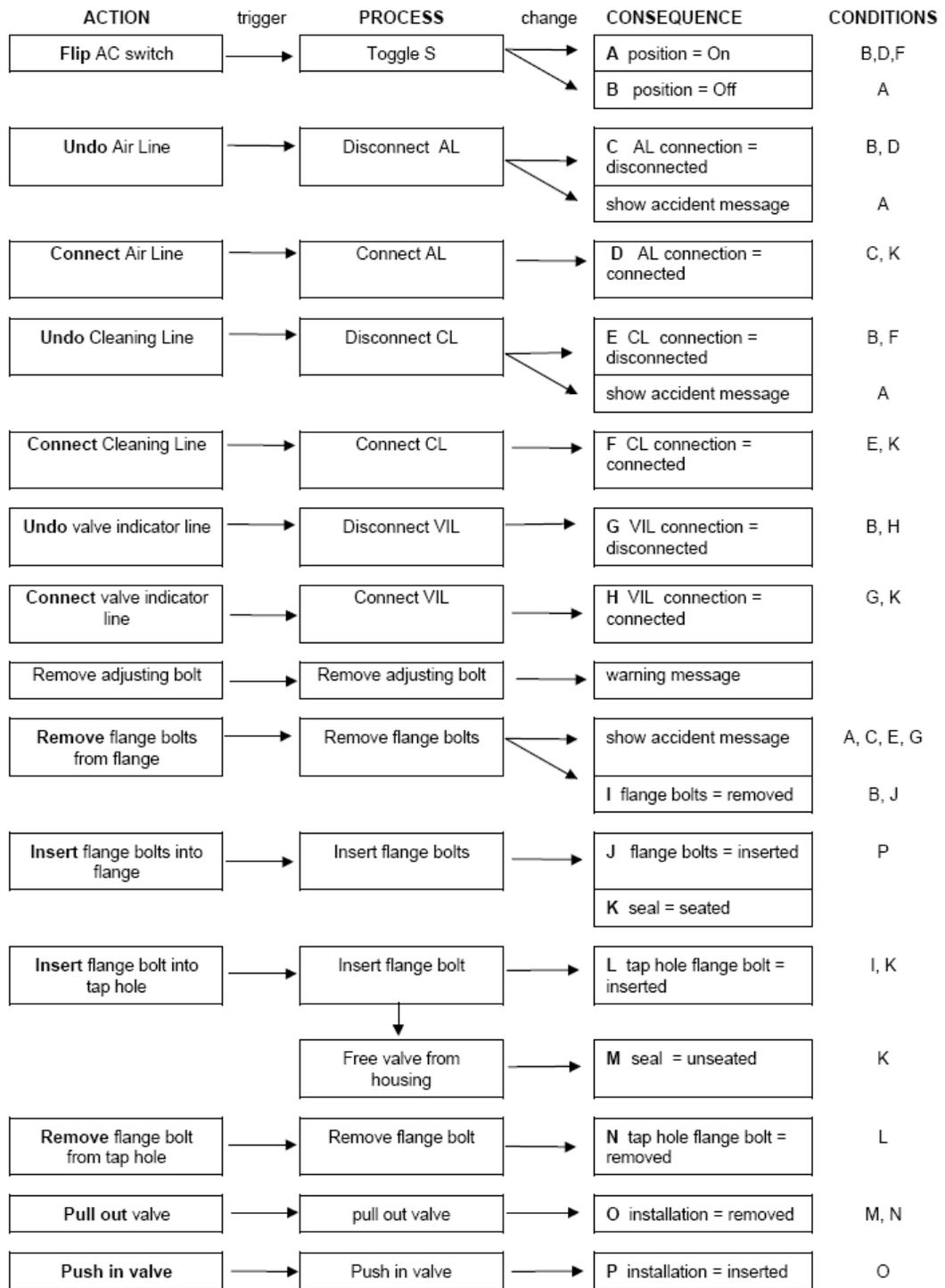| ACTION	trigger	PROCESS	change	CONSEQUENCE	CONDITIONS
Flip AC switch	→	Toggle S	→	A position = On	B,D,F
			→	B position = Off	A
Undo Air Line	→	Disconnect AL	→	C AL connection = disconnected	B, D
			→	show accident message	A
Connect Air Line	→	Connect AL	→	D AL connection = connected	C, K
Undo Cleaning Line	→	Disconnect CL	→	E CL connection = disconnected	B, F
			→	show accident message	A
Connect Cleaning Line	→	Connect CL	→	F CL connection = connected	E, K
Undo valve indicator line	→	Disconnect VIL	→	G VIL connection = disconnected	B, H
Connect valve indicator line	→	Connect VIL	→	H VIL connection = connected	G, K
Remove adjusting bolt	→	Remove adjusting bolt	→	warning message	
Remove flange bolts from flange	→	Remove flange bolts	→	show accident message	A, C, E, G
			→	I flange bolts = removed	B, J
Insert flange bolts into flange	→	Insert flange bolts	→	J flange bolts = inserted	P
				K seal = seated	
Insert flange bolt into tap hole	→	Insert flange bolt	→	L tap hole flange bolt = inserted	I, K
		Free valve from housing	→	M seal = unseated	K
Remove flange bolt from tap hole	→	Remove flange bolt	→	N tap hole flange bolt = removed	L
Pull out valve	→	pull out valve	→	O installation = removed	M, N
Push in valve	→	Push in valve	→	P installation = inserted	O

Figure 4 PEAnet knowledge structure for valve removal environment

Simulation Engine

PEAnet representation makes it possible to write a general simulation algorithm (sometimes called a simulation engine) which runs any learning environment constructed on PEAnet architecture. This algorithm monitors for an action (usually some mouse action on the screen); it interprets this action, meaning that it determines from the location of the mouse action which action has occurred; it then checks the conditions of the process triggered by this action; if the conditions are true it executes the process (changes the value of the property specified by the consequence) and displays the portrayal corresponding to the new value of the property; and if a process-to-process trigger is specified, it triggers the next process in the sequence. This instructional algorithm is written once and used over and over for different learning environments. Any situation, device, or phenomenon which can be represented by properties and their values can be represented in PEAnet architecture and run by the simulation engine.

The set of actions down the left side of Fig. 4 does not represent a linear sequence. For example, if the position of the air compressor switch is *off*, then the air line, cleaning line, or valve indicator line can be safely disconnected in any order. The learning environment allows the student to attempt to disconnect any of the lines even if the position of the air compressor switch is *on*, however, the system then displays an accident message indicating danger to the technician or damage to the valve as a result of this action. The student can also reverse any action. For example, the student can toggle the switch, reconnect a line, or reinsert the flange bolts at any time. Thus any action which the student could do in the real world environment is also possible to do in the learning environment, with some representation of the consequence of this action.

Exploration and Guidance

Without instructional overlay, exploration is the only learning activity enabled by this learning environment. The student can operate the device or explore the environment, experiencing consequences representative of those that would occur in the real world. Exploration represents a presentation alternative for a procedural or execute transaction.

It has been found that exploration without guidance is often insufficient for adequate learning. Exploration alone is inefficient. The student engages in trial-and-error behavior

and makes many wrong moves before coming across the appropriate sequence of actions to solve a problem. Often the student is unable, from exploration alone, to discover the most efficient sequence of actions to accomplish a given goal in the environment. From unguided practice alone most students are unable to discover the underlying conditions which would prepare them for prediction or trouble-shooting tasks. Hence, a learning environment without instructional overlay is only part of an instructional transaction and is therefore incomplete.

In order for a learning environment to be effective, it is necessary to provide different forms of learner guidance. One type of guidance is propaedeudic[4] instruction, such as learning the names, functions, and locations of the parts of the situation or device. Learning a procedure is more efficient if learner guidance takes the form of a demonstration in the learning environment itself, followed by scaffold[5] practice. To acquire prediction or troubleshooting skills the system must provide guidance in the form of explanations in the context of the student exploration. An explanation indicates those conditions which were met or not met when a given process executes or fails to execute. Finally, prediction and troubleshooting skills are developed when the student is required to predict the consequence of a given action or set of actions, or to find the conditions which prevented a given consequence or set of consequences from occurring (trouble-shooting). In the following sections we will describe each of these instructional transactions as they are implemented using knowledge objects in a learning environment.

IDENTIFY (COMPONENT) TRANSACTION GOAL

The student will be able to identify the name and location (with regard to some whole) of a given part of an entity (artifact, device, system, location, communication, etc.)

Knowledge Structure

In addition to the information slots (name, description, and portrayal), knowledge objects for component transactions require three additional slots: location, part-of, and has-parts. The location slot indicates the location of the portrayal with regard to some referent

---

[4] [7] *Propaedeutic* means to teach beforehand or to teach in preparation for some other learning activity
[5] *Scaffolded practice* is guided practice in doing a sequence of events where each successive trial withdraws the support until the student is performing the task in a completely unguided environment.

(the object to which the part belongs) knowledge object. The part-of slot contains a pointer to the referent knowledge object. The *has-parts* slots contains pointers to knowledge objects representing each of the component parts of the referent knowledge object.

The location slot has a parameter of mode with two values (graphical, temporal). If the portrayal of the knowledge object is text or graphic, then the location mode is graphical (which means that it can be located on a computer screen). If the portrayal is audio or video, then the location mode is temporal, meaning that the portrayal of the parts are some time segment of the portrayal of the referent knowledge object, for example, a segment of a video or a segment of a musical rendition.

In the example, each part of the valve is represented by a knowledge object that consists of a name (air hose connection), a description (see Figure 2), and a portrayal (that part of the graphic[6] in Figure 1 where the hose connects to the valve). The learning environment is a composite graphic. Each part of the valve is represented by its own picture (portrayal). Since each part of the system is its own portrayal (graphic), it also knows where it is on the screen. Hence each graphic contains information about its own location on the screen and can therefore highlight itself even if it is moved to a new location. The background of the learning environment is also a knowledge object. It contains pointers to all of the knowledge objects for each of the parts. Each part of the valve also has a pointer indicating that it is part of the learning environment representing the pump.

Presentation

The presentation mode for the IDENTIFY transaction is as follows: (a) Show the name and portrayal of the referent knowledge object. In our example, the name is in the title bar and the background for the valve including the pipe, the bottom plate of the valve, and the switch plate is the portrayal of the referent knowledge object. (b) Show the portrayal of each part of the referent knowledge object. In our example, the illustration of each part is the portrayal. For example, the valve itself, the flange bolts, the air hose connection, the air hose, etc. (c) If explore mode is enabled: on mouse-enter, show the

---

[6] The Learning Environment is composed of a composite graphic. Each part is represented by its own graphic. The entire set of graphics makes up the picture of the system seen by the student.

name of each part; on right click show the description of each part. (d) If lecture mode is requested "Tell me about some of the parts" : highlight each part in the selected item list, show its name, show its description. For temporal portrayals (video or audio) some graphic portrayal usually accompanies the temporal portrayal to identify the parts in time.

Practice

The practice mode for the IDENTIFY transaction is as follows: If *locate parts* is selected: present a part name, the student clicks on the part, provide right/wrong with correct answer feedback. If wrong, retain item in the list. If *name parts* is selected: highlight a part, present a list of part names, the student clicks on the name of the part, provide right/wrong with correct answer feedback. If wrong, retain item in the list. If *identify function* is selected: present a function description, the student clicks on the part, provide right/wrong with correct answer feedback. If wrong, retain item in the list.

Learner Guidance

During practice, if a student points to an incorrect part, the correct part is highlighted for the student. During practice, if a student selects or types the wrong name for a given part, the correct name is given.

Parameters[7]

Presentation and practice strategies can be controlled by a number of parameters. In a given transaction these parameters can produce a variety of different presentation or practice combinations. Some of the parameters for an IDENTIFY transaction include the following: show name (yes/no) and name mode (text/audio); show portrayal (yes/no) and portrayal mode (text/audio/graphic/video/ combination); show description (yes/no) and description mode (text/audio). By employing these parameters, a given part may be represented to the student in any of over 128 different combinations. In the example, the description was a combination graphic and description.

---

[7] A complete list of parameters for each of the transactions and all of their implications is considerably beyond the scope of this paper. See Merrill, Li, and Jones (1992) for more details.

EXECUTE (ACTIVITY) TRANSACTION

Goal

The student is able to execute a series of actions which lead to some consequence.

Knowledge Structure

The PEAnet structure for a learning environment enables the student to execute the activity. The PEAnet consists of a set of activities, each of which triggers a process which leads to some consequence (change in property value of some entity in the system). In the example, the PEAnet structure of Figure 4 is the knowledge structure for the activity of valve removal.

Inference Engine

PEAnet knowledge representation makes it possible to write an inference algorithm (inference engine) which can determine an appropriate path to a goal from any set of initial conditions (values of properties). The author or student indicates a goal for a procedure. A goal is a value for one or more properties in the system. The inference engine then determines which process sets the goal property to the goal value. It then determines action (trigger) for this process. If a condition for the process is true, the inference engine goes to the next condition. If a condition is false, the inference engine determines the process that will make this condition true. It then determines the action (trigger) for this second process. If a condition for this second process is true, the inference engine skips to the next condition. If this next condition is false the inference engine determines a third process that will make this condition true. This backward chaining continues until all the necessary processes and action (triggers) have been identified. A list of the actions thus determined, in reverse order, is a correct path to the goal.

Presentation

One type of presentation is a hands-off demonstration. In this type of demonstration the guide does the first action by automatically moving the cursor to the appropriate entity part and simulating a click, enabling the system to show the consequence; the guide then does the next action, etc., until the goal has been reached. Since it is passive, we feel that this type of presentation is less appropriate than the Simon Says demonstration described in the following paragraph. Simon Says requires the student to be actively involved in the

demonstration, focusing the student's attention on the action to be taken and the consequence of this action.

An appropriate presentation for teaching a procedure (sequence of actions) is a demonstration or Simon Says simulation. The inference engine determines a path (sequence of actions) from the initial state of the system to the goal. The guide then directs the student to execute each of these actions in turn with the direction "Do <action name>." If the student does this action its consequence is executed and the guide directs the student to do the next action. If the student does some other action, the guide presents a message, "That is not the <entity part on which the action should have been taken>. Try again." This guided demonstration is continued until the goal has been reached.

A Simon Says demonstration is sometimes called *guided practice.* We feel that this is a misnomer: that this type of learning experience is not practice but rather *active presentation.* In our view a presentation need not be passive. Requiring the student to carry out each step by following directions focuses the student's attention on the part and action involved in the step.

Practice

We identified two additional levels of practice. *Next step* practice is the same as Simon Says demonstration, except the guide does not indicate the action to be taken but merely gives the direction: "Do the next step." The student must remember which step is next and then do the action. If the student does a different action, the same reminder is given as with the Simon Says demonstration, that is, "That is not the <name of part on which the action should have been taken>."

In *You-do-it* practice, the system is put in open-ended mode, enabling the system to execute any consequence to any process trigged by any action within the limitations of the system. However, the guide indicates the goal of the performance and directs the student to take those actions necessary to accomplish the goal. The guide also directs students to click on a *finished* button when they believe they have accomplished the goal. When students click on the *finished* button, they are told whether or not they have correctly finished the task (accomplished the goal). The guide then shows the student the path (sequence of actions) that the inference determined and shows the student's path (actual actions taken by

the student). Unnecessary actions on the part of the student are highlighted to facilitate the student's comparison of his or her path with the guide's path. In systems where there are multiple paths to the goal the student's path may not be the same as the guide's but may still accomplish the goal. The system recognizes that the goal has been correctly accomplished. The system records the student's actual path and the number of steps required for the student to accomplish the goal.

For complex systems additional practice is desirable. A number of different problems can be defined. A problem is a different set of starting conditions. Selecting a different problem resets the starting conditions to some predetermined values. Such divergent practice facilitates transfer to configurations of the system not encountered as part of the instruction. Additional practice is also desirable when a procedure can be executed with a variety of different, but similar, systems. Practice with a number of these divergent systems will facilitate transfer of the activity to yet other systems not encountered by the student during the instruction.

Learner Guidance

The scaffolding nature of the practice from a *Simon Says* demonstration to a *You-do-it* practice is one form of learner guidance. This successive progression from highly guided interaction to unguided interaction has been recommended by a number of different instructional-design theorists and has been supported by empirical investigations to facilitate the learning of procedural skills.

Providing an explanation to students as they learn a sequence of actions has also been recommended and shown to facilitate the acquisition of procedural skill. PEAnet knowledge structure makes it possible to build a system that can automatically provide explanations under a variety of circumstances. In an explore mode the student can request an explanation from the guide. In a *Simon Says* demonstration mode or *Next step* practice mode, the learner can turn on the *explanation*, which is then displayed as each step in the procedure is executed. In a *You-do-it* or *performance* practice mode the explanation can be turned on and displayed as the student executes each action.

An explanation has two parts: what happened? and why? In terms of the elements of knowledge objects, what happens is the consequence or change of property value caused

by the process. An explanation template enables the guide to provide this what-happened explanation. One template is as follows: When the student requests an explanation, the guide present this message when the process executes: "When you <action name> the <property> of <entity who owns property> is changed to <value>," and this message when the process does not execute: "when you <action name> nothing happens." A system may have different explanation templates for different kinds of values and different operations for changing those values but these details are beyond the scope of this chapter.

The why part of the explanation is implemented by presenting the condition(s) that had to be satisfied in order for the process to execute. The why part of the explanation presents a text template of the following form when the process executes: "This happens because the <value> of <property> is <value>," or the following when the process does not execute: "This happens because the <value> of <property> is not <value>."

In the example, suppose the student attempts to remove the valve before removing the flange bolt from the tap hole. The explanation presented by the guide reads as follows: "When you <pull out double seat valve> nothing happens. This happens because the <location> of the <flange bolt> is not <removed from tape hole>."

INTERPRET (PROCESS) TRANSACTION GOAL

Given a set of conditions, the student is able to predict the consequence of an event. Or given a consequence (expected or unexpected) the student is able to identify the conditions which were present in order for this consequence to occur. When an unexpected consequence occurs (an error), then finding the precipitating conditions is called *trouble shooting*.

Knowledge Structure

The knowledge structure required for an interpret transaction is a set of process rules consisting of conditions and consequences, a goal to be accomplished (value for some property or set of properties), and a set of specific problems (initial conditions). Some of the problems may contain faults. A fault is an incorrect condition. That is, when a faulted process executes, an unexpected consequence occurs because of the incorrect condition.

An incorrect condition can represent a damaged component, a missing component, or a control set to an inappropriate value.

The same PEAnet knowledge structure described for the explore interaction is required for an INTERPRET transaction. It may be necessary to include additional conditions representing faults. Selecting a problem is an action which triggers a process which changes the initial conditions of the system to some predetermined values.

Presentation

The student is allowed to play with or explore the learning environment to "see what happens if…". An explanation (a form of learner guidance) is available that indicates the consequence of each action ("what happened") and the conditions that were satisfied or unsatisfied ("why"). If the process applies to a number of different situations, then a number of different scenarios (variations on the learning environment) are provided to the student, and the student is allowed to "play with" these variations. In some situations the student is provided with a "control panel" that allows them to set the values of some of the properties of the system. In this way the student can perform experiments by observing the consequences of different conditions (property values) and receiving the explanation for these consequences. One advantage of a learning environment is that there need not be a distinction between conditions that students can change as a result of their actions (action triggers process, which changes a condition or consequence) and conditions which cannot be changed by student actions. We can always give a student some control action (an action not available in the real world) that allows them to experiment with the system. (An example is a gravity control that increases or decreases gravity.)

Practice

The student is presented with a specific problem in the learning environment. The student is asked to observe the conditions of the device or system and to predict one or more consequences. In the system described, the student is provided a list of properties and allowed to select the value for each property that constitutes the consequence of a given action. One or more of the conditions given may represent a faulted condition. A variation is to provide the student with a control panel and direct them to set the conditions (property values) which will lead to a specified consequence. The consequence may be one that

would result from a faulted condition. The prediction is confirmed by allowing the system to execute and showing the consequence of the execution. In complex systems the student may have to trace the execution back through several events to find the condition(s) which caused the observed consequence. Or the student may need to execute several events of the system to see the consequence that results from the conditions.

Learner Guidance

During the presentation or exploration the conditions necessary for a consequence of a given event are made clear to the student. Often the best guidance is to allow students to ask for an explanation of an unexpected consequence during their exploration or the system. This explanation identifies the conditions that caused the consequence. During practice the student's predictions or trouble-shooting are confirmed by executing the system with explanations of what occurred during each step of the process and why.

Adaptive Instruction

Perhaps one of the most exciting possibilities for the use of knowledge object architecture for instructional-design is the possibility for truly adaptive instruction. An instructional strategy is an algorithm An algorithm can be prespecified and preprogrammed. In addition an instructional algorithm can include a number of parameters, the different values of which control the way the strategy promotes interaction with the student. Changing a strategy parameter value changes the nature of the interaction. An adaptive system would include a set of expert system rules relating student parameters to instructional strategy parameters. When a student parameter changes, the expert system would then change the parameter of the strategies, and the subsequent interaction with the student would change. An adaptive system would include a system for monitoring student parameters (level of motivation, level of interest, performance, and other parameters). When the values on these student parameters change then the system dynamically changes the values of the strategy parameters, thus adapting the system to the individual student.

The knowledge object architecture together with its simulation engine and inference engine makes it possible to build a learning environment in which students themselves are objects in the learning environment. Like other objects in the learning environment, the "student entity" has properties and property values. These property values are changed by

processes triggered by actions or other processes just like all other entities in the system. Thus, when the properties of the student entity change, the actions that are available to the student change. The learning environment thus adapts to the student or interacts with the student in a dynamic way.

There is not space here to elaborate on adaptive systems. A subsequent paper will describe a theoretical model for adaptive instruction.

SUMMARY

In this chapter we have concentrated on the component methods of ITT . We have suggested that a more precise representation of the knowledge to be taught in the form of knowledge objects increases the precision with which instructional strategies can be described. We have indicated that instructional strategies can be described as methods for manipulating the elements of knowledge objects. This architecture enables the specification of executable knowledge, making possible tutorial and experiential instruction from the same knowledge representation.

In this chapter we have described knowledge objects. We have described how knowledge objects are used to define a number of instructional strategies including: learning environments (exploration), IDENTIFY transactions, EXECUTE transactions, and INTERPRET transactions. We have illustrated one implementation of these strategies in an instructional learning environment.

In our previous presentations of CDT we have provided prescriptions for best-case strategies for different kinds of instructional outcomes. In this chapter we could have recast these prescriptions in terms of strategy algorithms based on knowledge objects, but we chose instead to present the architecture of the system rather than the prescriptions. This chapter is a presentation of the foundation for instructional-design based on knowledge objects but is not a complete presentation of the theory and does not include all of the items that were suggested by the editor of this volume.

BIBLIOGRAPHY

Gagné, Robert M. (1965). The Conditions of Learning. New York: Holt, Rinehart and Winston.

Gagné , Robert M. (1985). The Conditions of Learning and Theory of Instruction. Fourth Edition. NewYork: Holt, Rinehart and Winston.

Jones, M.K., Li, Z. & Merrill, M.D. (1990). Domain knowledge representation for instructional analysis. Educational Technology, 30(10), 7-32.

Li, Z. & Merrill, M. D. (1990). Transaction shells: a new approach to courseware authoring. Journal of Research on Computing in Education, 23(1), 72-86.

Merrill, M. David (1983). Chapter 9 Component Display Theory. In Charles M. Reigeluth (Ed.) Instructional-Design Theories and Models. Hillsdale, NJ: Lawrence Erlbaum Associates.

Merrill, M. David (1987). Chapter 7 A Lesson Based on Component Display Theory. In Charles M. Reigeluth (Ed.) Instructional Design Theories in Action. Hillsdale, NJ: Lawrence Erlbaum Associates.

Merrill, M. David. (1997). Learning-Oriented Instructional Development Tools. Performance Improvement, 36(3), 51-55.

Merrill, M. D., Li, Z. & Jones, M.K. (1991) Instructional Transaction Theory: an introduction. Educational Technology. 31(6), 7-12.

Merrill, M. David, Jones, Mark K., & Li, Zhongmin. (1992). Instructional Transaction Theory: classes of transactions. Educational Technology. 32 (6), 12-26.

Merrill, M. David, Li, Zhongmin, & Jones, Mark K. (1992) Instructional Transaction Shells: responsibilities, methods, and parameters. Educational Technology. 32 (2), 5-27.

Merrill, M. David & ID2 Research Team (1993). Instructional Transaction Theory: knowledge relationships among processes, entities, and activities. Educational Technology, 33 (4), 5-16.

Merrill, M. David (with David G. Twitchell, Editor), (1994). Instructional Design Theory. Englewood Cliffs, NJ: Educational Technology Publications.

Merrill, M. David & ID2 Research Team (1996). Instructional Transaction Theory: Instructional Design based on Knowledge Objects. Educational Technology, 36 (3), 30 37.