N96- 12958

5 P 62

p. 12

Building an Intelligent Tutoring System for Procedural Domains

By Andrew Warinner, Diann Barbee, Larry Brandt, Tom Chen, and John Maguire
Global Information Systems Technology, Inc.
1800 Woodfield Drive,
Savoy, Illinois 61874

Introduction

Jobs that require complex skills that are too expensive or dangerous to develop often use simulators in training. The strength of a simulator is its ability to mimic the "real world", allowing students to explore and experiment. A good simulation helps the student develop a "mental model" of the real world. The closer the simulation is to "real life", the less difficulties there are transferring skills and mental models developed on the simulator to the real job. As graphics workstations increase in power and become more affordable they become attractive candidates for developing computer-based simulations for use in training. Computer-based simulations can make training more interesting and accessible to the student.

Unfortunately, good simulations do not necessarily make good trainers. One of the main tenets of most current learning theory is that the development of new knowledge is greatly constrained by what an individual already knows[1]. Simulations may require complex skills that are difficult to develop individually in sophisticated simulation. The student may not be able to use the simulation until the prerequisite knowledge and skills have been learned. Computer simulations are more flexible than dedicated, "task specific" simulations since they can simulate situations that are not strictly "realistic" but can reduce the complexity of the simulation in order to develop basic skills and concepts.

Although a simulation is a learning environment, it offers the learner no instructional assistance. We believe learning is greatly enhanced when instructional techniques are added to a simulation. For the past three years we have been exploring the challenges of incorporating "intelligent tutoring systems" (ITS) into computer-based simulations. Developing an intelligent tutoring system for a simulation really requires the development of two cooperating expert systems: a *domain* expert system that serves as a basis for evaluation of the student and an *instructional* expert system that can compare the student to the domain expert and prescribe training.

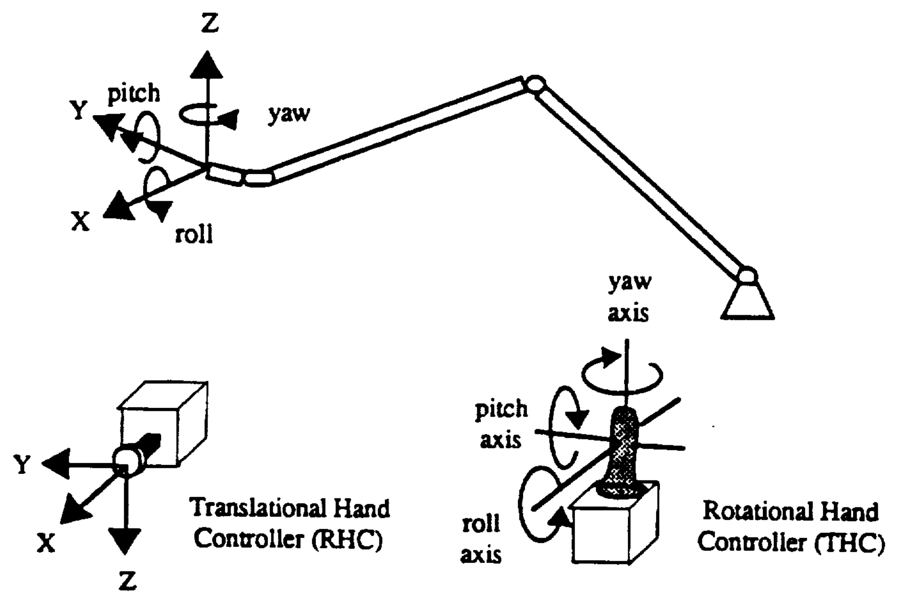

Z
Y
pitch
yaw
X
roll
yaw axis
pitch axis
Y
X
Z
Translational Hand Controller (RHC)
roll axis
Rotational Hand Controller (THC)

Figure 1. Controlling the RMS is Orbiter Unloaded Mode

The Domain: The RMS

The Remote Manipulator System (RMS) is the mechanical arm of the Payload Deployment and Retrieval System (PDRS) of the Shuttle. It is used to grapple a payload stowed in the Shuttle's cargo bay and lift it into orbit or grapple a payload in orbit and berth it in the cargo bay. Like a human arm, the RMS has three joints, a shoulder, elbow, and wrist, each with varying degrees of freedom (possible directions of movement). The arm is attached at the shoulder to the longeron of the Shuttle bay and is over fifty feet in length. At the end of the RMS is "end effector". The end effector used to grasp and hold the payload. Like a human arm, the RMS has physical limits on the roll, pitch and yaw of each joint. The RMS has movement limits imposed by a computer that monitors the RMS to reduce the possibility of damage. The RMS can be moved into positions where it loses one of its degrees of freedom (i.e. when movement of a joint in a specific direction becomes impossible). These configurations are called "singularities". The operator must reposition the RMS when it is in a singularity to regain its freedom of movement.

The RMS operator controls the the arm from the rear of the Shuttle cockpit. It is controlled with two hand controllers: a "translation hand controller" (THC) and a "rotational hand controller" (RHC). The operator can view the payload and RMS from windows or on a closed circuit TV (CCTV) from several cameras positioned about the Shuttle.

The RMS has several modes of operation. The RMS can be entirely controlled by the Shuttle's general purpose computer (GPC). The GPC can assist the shuttle operator in operating the arm or the operator can control the arm without computer assistance. These different modes of operation use different coordinate systems to describe the position of the RMS, the Shuttle, and the payload. The different modes also change the effects of the hand controllers on the position of the RMS (see Figure 1).

Successful operation of the RMS requires motor skills, complex cognitive skills, and knowledge of the mechanics of the RMS. To master the RMS the operator must learn the limits of the RMS and how to control its different modes. An understanding of the different coordinate systems and the ability to visualize arm and payload movements in space relative to the Shuttle are also important for successful RMS control. Operators must learn to manipulate the arm efficiently and safely.

Over the past three years, NASA has developed a computer-based simulation of the RMS called the Prototype Part Task Trainer (P2T2). Running on a color graphics workstation, P2T2 simulates the RMS and its different modes of operation using the same algorithms as the GPC. P2T2 simulates the different camera views available from the CCTV as well as the RMS control panels. P2T2's hand controllers are exact replicas of the THC and the RHC on the Shuttle.

Our goal is to embed an intelligent tutoring system into P2T2 to make it a more effective training device. The ITS/P2T2 will be a stand-alone trainer capable of teaching the domain of RMS operation. We will use CLIPS as the inference engine of of the ITS. CLIPS has several advantages over other inference engines:

- ability to be embedded in other applications, P2T2 in our case

- CLIPS is written in C and runs under UNIX®, P2T2 is written in C and runs under a variant of UNIX

- source code is provided, allowing us to make special modifications

Since we must build our ITS into P2T2, CLIPS ability to be embedded is important. Performance is another critical concern. Our ITS needs real-time performance in order to monitor and instruct the student.

Intelligent Tutoring Systems

The primary difference between intelligent tutoring systems and more traditional computer-based training is the "student model", a representation of the skills and knowledge possessed by the student. An intelligent tutoring system contains a student model, a computer-based training lesson does not.

The instructional expert uses the student model to gauge the student's progress and prescribe instruction. The domain expert compares the student to the "correct" performance it generates and provides the results to the student model. Since the student model is the used by both the instructional expert and the domain expert, the student model must have a representation that is accessible to both experts. Figure 2 illustrates how the two expert systems in the ITS act on the student model.

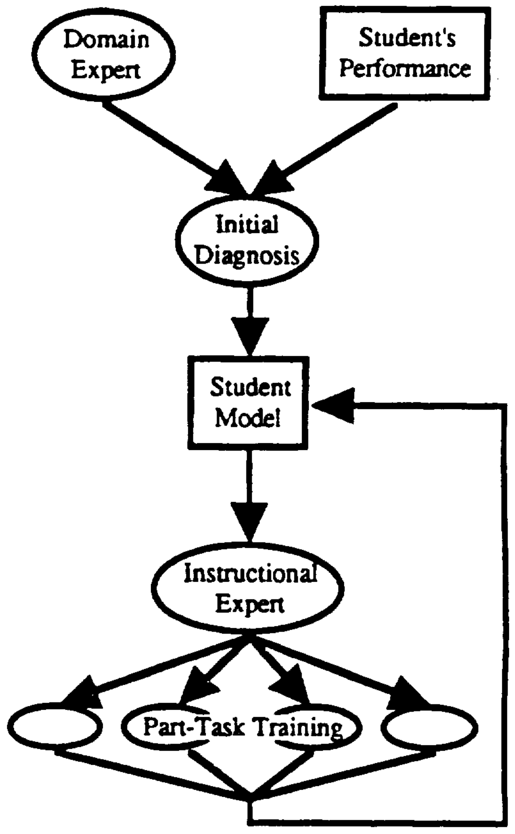


Figure 2. The Student Model, the Domain Expert and the Instructional Expert

We have chosen to represent the student model as a hierarchical network of skills and concepts necessary to master the RMS. Each skill or concept can have supporting subskills and subconcepts. A subskill or subconcept may support several skills or concepts. We

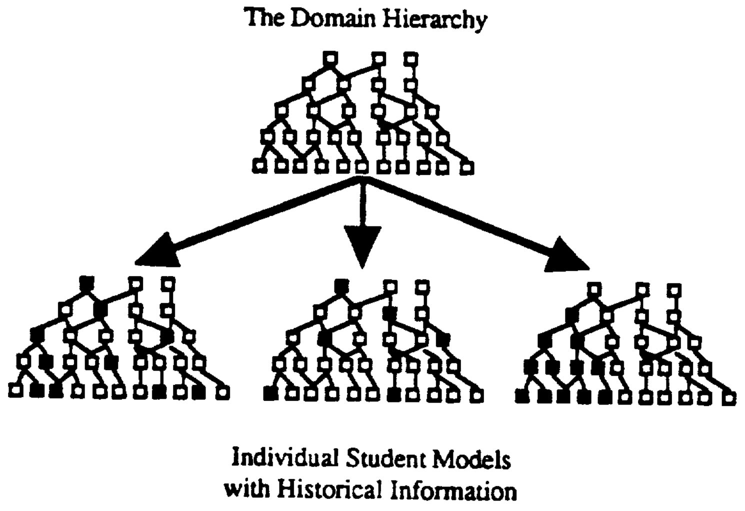


Figure 3. The Domain Hierarchy and the Student Models

call this taxonomy of the RMS domain the "domain hierarchy". Each skill or concept is represented by a node in the domain hierarchy. The student model is a copy of the domain hierarchy that stores information about the student's mastery or misuse of each skill or concept. Figure 3 illustrates the domain hierarchy and the student model.

The domain hierarchy/student model is a good representation for both diagnosis and instruction. Part-task training can use the hierarchical taxonomy of the domain to organize instruction. Diagnostically, the student model functions as a decision tree to which we apply algorithms drawn from electronic fault isolation. The diagnostic and instructional functions of the student model will be explained in more detail.

The Domain Expert

The domain expert provides the means to analyze the student. It must "understand" its domain. The domain expert must solve problems as an expert as well as be able to understand the student's actions and compare them to its solution. We have found the best representation for the domain expert is the "procedural network". Procedural networks have been used before in intelligent tutoring systems, for example, the "BUGGY" ITS developed by Brown and Burton[2]. The procedural network is a powerful representation of how the skills and tasks of the domain are related. Procedural networks are a good representation for an ITS that tutors a procedural or task-oriented domain[3]. Briefly, the advantages of a procedural network are:

- goal-based representation of the task or procedure allows for a flexible evaluation of student performance

- real time evaluation of the procedure

- multiple levels of abstraction in the procedure

- mechanisms for representing the partial ordering of procedures

- a representation of the "world" as it relates to the procedure

Procedural networks can be constructed dynamically. Our ITS will not dynamically construct its procedural network for two reasons. First, we have chosen to restrict knowledge acquisition to a small set of tasks in the RMS domain. Second, the dynamic construction of procedural networks uses difficult techniques such as plan criticism and plan optimization. Dynamically constructed procedural networks might contain flaws that would limit their usefulness during student evaluation. Our architecture does not preclude the dynamic construction of procedural networks if they are needed in the future.

Hierarchical Reasoning in Procedural Networks

The hierarchical nature of procedural nets makes them ideal for reasoning about the procedure at different levels. Student diagnosis can measure skills and performance at different levels of the procedure. For example, we might want to measure an overall quantity like the time to perform a section of the procedure. The hierarchical nature of the procedural network allows us to measure skills at different levels in the procedural network without examining and interpreting the individual actions that accomplish that section of the procedure.

Flexible Framework for Plan Recognition

One of the most difficult tasks in procedure evaluation is understanding the student's progress through the procedure. Often procedures contain some flexibility in the order of steps or tasks performed. Procedures can offer opportunities for the student to correct his or her mistakes and continue. A step-by-step comparison of the student and the expert's solution is too rigorous. If the procedure contains tasks that can be performed in any order, we can't rely on a step-by-step ordering of the student's actions for evaluation. The procedural network's orsplit nodes are a good representation for such flexible plans. The procedural network's andsplit nodes can represent the strict ordering of procedure steps.

For example, suppose a section of the procedural network contains this procedure of independent tasks:

1. Reset the widget A (press button 1)
2. Turn on widget B (turn knob 1 to "on")
3. Prepare widget C (accomplished by subprocedure)
    3.1. Set gizmo 1 (turn knob 2 to "5")
    3.2. Turn off gizmo 2 (turn switch 1 to "off")

Suppose that the widgets and gizmos are independent mechanisms: manipulating one widget does not affect the operation of any of the others. If the procedure was executed in strict sequence it would result in the following sequence of actions:

1. press button 1
2. turn knob 1 to on
3. turn knob 2 to 5
4. set switch 1 to off

But suppose the student executes the actions in this order:

1. set switch 1 to off
2. press button 1
3. turn knob 2 to 5
4. turn knob 1 to on

The procedural network can interpret this sequence of actions as accomplishing the procedure even though the actions are not in strict order. See Figure 4 for an illustration of this procedure.

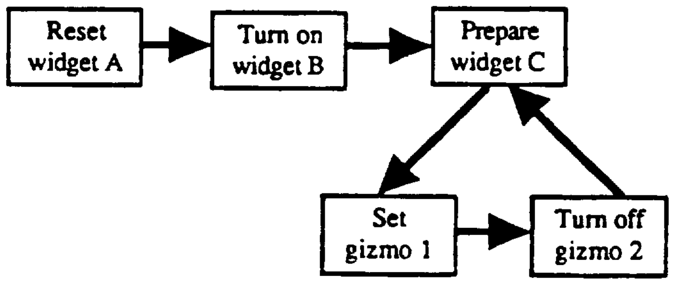


Figure 4. A procedural network example

Procedural Networks and Real-Time Evaluation

Another advantage in using procedural networks as a representation is that they can be used to evaluate the procedure as it is performed. Real-time evaluation is needed if some kind of coaching feedback is provided to the student. Student evaluation becomes a process of parsing the student's actions and comparing them to the procedural network. This can be done in a top-down fashion to the necessary level of detail. The state of the procedural network at any point in time is a complete description of the state of the world as well as the state of the procedure. As the student moves through the procedure, the interpretation of his or her actions is based on how they changed the state of the world.

Representing the World State

As mentioned above, the procedural network is not only a representation that describes the procedure but also the state of the world. The procedural network describes how each step of the procedure affects the state of the world. This description of the procedure allows reasoning by the modules of the ITS on the effects and relationship of parts of the procedural network.

Procedural Networks

Procedural networks were first characterized by Sacerdoti[4]. They are closely related to augmented transition networks and generalized and-or graphs. The procedural network is ordered by its links to among nodes. Nodes may have predecessors, successors, a parent and children. The successor and predecessor links order the procedure. The parent and child links denote subprocedures that must be executed to achieve the effects of the parent procedure. The parent and child links allow the procedural network to be ordered hierarchically. The procedural network is composed of four basic classes of nodes described below.

Procedure start and procedure end nodes are delimiters of the procedure. They are used for both procedures and subprocedures.

Goal nodes and subprocedures organize the procedural nets hierarchically. Goal nodes are accomplished by subprocedures that are linked as children. In the example illustrated in Figure 4 the node "Prep widget C" is a goal node that is accomplished by the subprocedure "Set gizmo 1" and "Gizmo 2 off" (note: the procedure start and procedure end nodes have been eliminated from the figure).

Andsplit and andjoin nodes delimit a collections of steps which may be performed independently. The andsplit and andjoin nodes themselves delimit the independent steps.

Orsplit and orjoin nodes are similar to the andsplit/andjoin nodes. They delimit a set of steps only one of which must be performed successfully. The orsplit and orjoin nodes delimit the steps.

Node effects are lists of effects on the world state. They represent the changes caused by completing the procedure step. The node effects will change machine values, positions, and other world state information.

Link predicates are used to control branches of the procedural network based on the state of the world. Since the procedural network is not constructed dynamically, link predicates enable or disable branches of the procedural network. For example, a branch for an error correction procedure may be enabled or disabled depending on the state of the equipment.

Procedural ordering links are used to represent ordering information not captured by the successor and predecessor links. The procedural ordering links are used to express ordering of the procedure not required by the machine states. The procedural ordering information is kept separate from the world state representation.

Comparing the Expert and the Student

As the student performs the procedure, the domain expert monitors his or her progress with the procedural network. When the student has finished the procedure or the instructional expert has intervened, the domain expert can refer to the procedural network to see what portions of the procedure were completed correctly and what portions were not completed correctly. The domain expert must now assess the causes of the student error and update the appropriate skills and concepts in the student model.

The domain expert must now translate the results of the procedural network into information about specific skills and concepts in the student model. The procedural network lends itself to a classification of student errors[5]. This classification uses the structural and world state information represented in the procedural network. Student errors fall into four classes:

- problem violations
- irrelevant procedures
- incorrect procedures
- ordering violations

Invalid action - This is an action that the student has taken that is not valid anywhere in the procedural network. Since the procedural network characterizes all possible paths through the network and all the possible actions that might be taken somewhere in the procedure, the domain expert can detect any action that does not fall on a path.

Problem violation - A student may take actions that are appropriate to achieve a goal but are inappropriate for the initial state of the world.

For example, suppose we have a procedure:

```
1. Power up the widget (goal)

(if widget is type A)
    1.1 Set power switch to "on"
    1.2 Press widget reset button

(if widget is type B)
    1.3 Set widget dial to "0"
    1.4 Set power switch to "start"
    1.5 Press widget reset button
    1.6 Set power switch to "on"
```

If we told the student that the widget is type A and he performs any of the steps 1.3 - 1.5 he has made a "problem violation".

Irrelevant procedure - Since we are not dynamically constructing the procedural network, we will use link predicates to disable unnecessary parts of the procedural network. Domain expert can detect if the student attempts to execute these disabled branches and report them as "irrelevant plans".

For example, suppose we have the following procedure:

```
1. Prepare gizmo (goal)

(if gizmo status is "error")
    1.1 Set gizmo power button to "off"
    1.2 Set gizmo power button to "on"
    1.3 Press gizmo reset button

(if gizmo status is "ok")
    1.4 Press gizmo button 1
    1.5 Press gizmo button 2
    1.6 Set gizmo switch to "on"
```

The "if" statements represent link predicates that enable or disable branches in the procedural network. If the student attempts to perform the steps of the error subprocedure, the domain expert will recognize them as "irrelevant plans".

Incorrect procedure - If the student omits a step in a procedure, the domain expert can detect this as an unsatisfied node in the procedural network. Domain expert will classify the missed step as an "incorrect procedure".

Ordering violation - We have added the procedural ordering links to the procedural network to represent ordering of the procedure not required by the world state. Domain expert will use these procedural ordering links to detect violations in the ordering of actions that are not mandated by node effects.

For example, suppose we have the following procedure:

1. Prepare the widget (goal)
   1.1 Set switch to "A"
   1.2 Press button 1
   1.3 Turn dial to "5"

Suppose the switch, button, and dial are independent of each other; the operation of one does not affect the others. This would be represented in the procedural network as an andsplit/andjoin branch. We can add procedural ordering links to represent the fact that we want the steps 1.1, 1.2, and 1.3 performed in strict order. Suppose the student performed the actions in this order:

1. Pressed button 1
2. Turned dial to "5"
3. Set switch to "A"

Domain expert can diagnose this as a ordering violation error but it will not classify it as an incorrect procedure error since the student has not violated the andsplit/andjoin construct in the procedural network.

Error Evaluation

After the domain expert has classified the errors observed in the procedural network, those errors must be mapped to corresponding skills and concepts in the student model. Each step in the procedural network has pointers to skills and concepts necessary to successfully perform that step in the procedure. As we noted before, the error classifications can help interpret the mistakes observed in the procedural network. In addition, we can use the historical information in the student model to assist in the diagnosis. Each step in the procedural network has links to several skills and concepts in the domain hierarchy:

- "knowledge" nodes that represent that the student is aware of the procedure step

- "condition" nodes that represent the student's knowledge of the conditions when the procedure step should be performed

- "skill" nodes that represent the satisfactory performance of the procedure step

- "effects" nodes that represent the effect of the procedure step on the state of the world

For example, suppose the domain expert detected a "problem violation" error. There are several plausible explanations for this error:

- the student is not aware of the conditions under which the procedure step should be performed

- it was a transient error; the student ignored or misinterpreted the conditions

- the student is ignorant of the effects of the procedure step on the state of the world

Each of these plausible explanations are represented by nodes in the domain hierarchy. To some extent the explanations are mutually exclusive. How does the domain expert choose between them? The domain expert can use the historical information from the student model. Continuing our example, suppose the student model shows that the student has never been exposed to concepts that represent "knowledge" of the procedure step, the domain expert can rule out the possibility that it was a transient error. On the other hand, if the student model shows that the student was familiar with the procedure step but has not used the procedure step in some time. The domain expert will favor the explanation that it was a transient error.

We have found that an analysis of the procedural network can provide information about only a subset of the skills and concepts in the student model. The domain expert can only infer information by observing the student. But the student model contains high-level abstractions and low-level skills and whose use cannot be observed directly. For example, a high level concept like "safety" cannot be associated with a single procedure step. An abstract concept "knowledge of RMS coordinate systems" would be difficult to deduce from simply observing the student. In general, the domain expert is able to draw conclusions about intermediate skills and concepts in the student model[6].

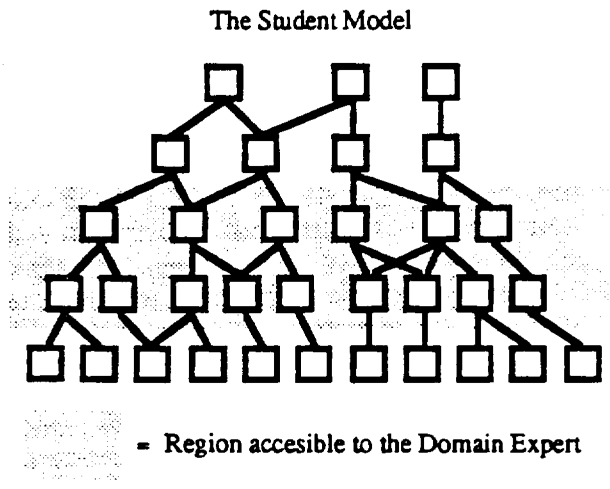


Figure 5. Regions available to the Domain Expert's diagnosis

The Instructional Expert

So far we have discussed the diagnostic aspects of an intelligent tutoring system. The diagnostic functionality is only half of an ITS, the other half is its tutoring functionality. An ITS can be viewed as a expert system compares the "expert model" of the domain to the "student model" that represents a novice student. The ITS then determines "operations" that will transform the student model to match the expert model.

Once the domain expert has updated the student model based on the result of its diagnosis, the instructional expert takes over. The instructional expert must examine the state of the student and apply remediation to the weaknesses it finds there.

We have chosen to provide tutoring to the student by means of part task training. Part task training is based on a systematic analysis of the instructional domain. The analysis identifies the skills, strategies, and knowledge necessary for expert performance. It also identifies the hierarchical relationships among the skills and knowledge. As an example of this, Figure 6 illustrates a part-task analysis of the RMS domain. The skill "Payload Deployment" is composed of the skills "Payload Release", "Payload Unberthing", "Move to Grapple Position", "Grappling the Payload", and "Ungrappling the Payload". The skill "Payload Deployment" is an "integration" skill. It requires mastery or "integration" of some subskills. The subskills may themselves be decomposed into other skills.

Once an analysis of domain is complete, training is designed to develop proficiency in the skills and concepts found in the domain hierarchy. A part task is designed to teach exactly that skill or strategy. When the student is proficient in all the subskills of an integration skill then the student can be trained in the integration skill[7].

Both the diagnostic functionality and the instructional functionality can exploit the hierarchical organization of the expert domain. The hierarchical organization can be

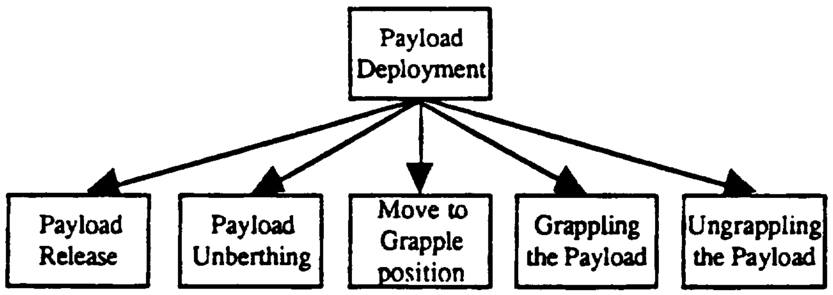


Figure 6. Sample Part-Task Analysis of the RMS Domain

used as a sort of "decision tree" using the historical information in the student model. The hierarchical structure of the student model is used to organize and relate the skills and knowledge of the domain for the instructional expert.

Final Diagnosis and Instruction

At any given time, the student may have some misconceptions or lack proficiency in skills in the domain. How does the instructional expert decide when and what can be tutored? The question of when the student should be tutored can be answered by the historical information stored in the student model. We have adapted algorithms from electronic circuit fault diagnosis to answer the questions of what should be tutored and when it should be tutored.

The fault isolation algorithms utilize the hierarchical structure of the student model/domain hierarchy. There is a certain amount of "overlap" in the hierarchical structure of the student model. Some subskills are required by several skills. The fault isolation algorithms can use these interrelationships to help us distinguish the "source" of error from its "symptoms" in the student model. Suppose we know that skill A^1 and A^2 have the subskills B^1, C^1, and C^2 in common and the domain expert has determined that student has misused them (see Figure 7). The problem may be in the skills A^1 and A^2 or in their supporting subskills. Our fault isolation algorithms attempt to explain the deficiencies by looking for areas that the deficiencies have in common. These common areas might be the real cause of the deficiencies. In our example, the fault isolation algorithm would consider the skills B^1, C^1, and C^2 as the real source of the student's misconceptions and the skills A^1 and A^2 as symptoms. The fault isolation algorithms attempts to find the simplest explanation that accounts for the most errors in the student model. Furthermore, they can recommend a skill to be tested that will eliminate the most uncertainty about where the real source of error lies in the student model.

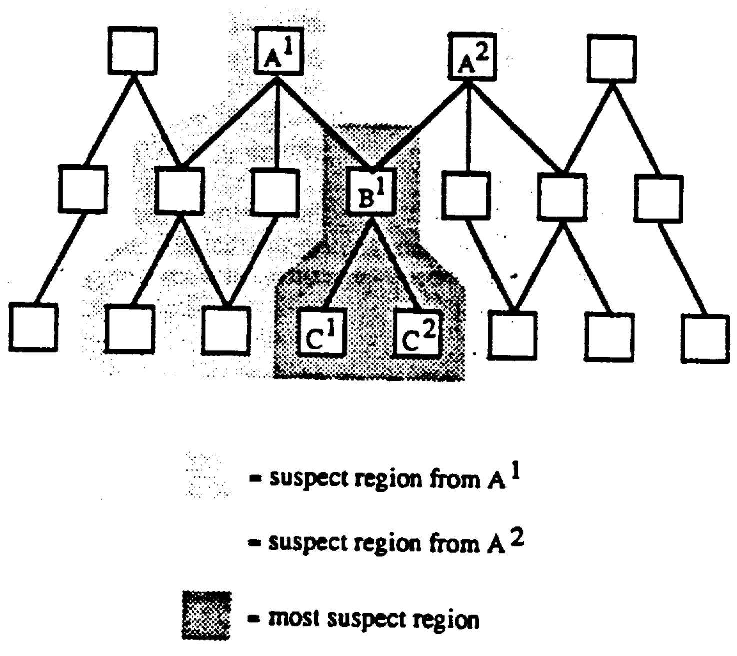


Figure 7. Applying fault isolation techniques to the Student Model

The fault isolation algorithms provide:

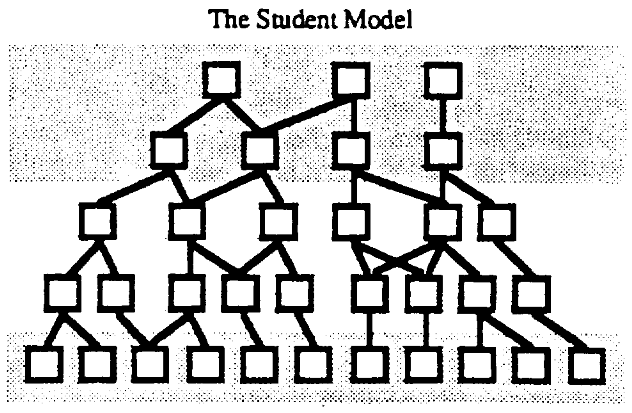
- a skill or concept that it has isolated as the source of the student's misunderstanding

- or a region in the domain hierarchy where errors are located and a specific skill or concept that is the mostly likely source of error

The instructional expert must now determine which part-task training will remedy the deficiencies observed in the student. One of the functions of the domain hierarchy is to serve as a map to the part-tasks. Given a set of skills and concepts misused by the student, the instructional expert can find a set of part tasks that will instruct the student.

The instructional expert must organize the part-task training it presents to the student. The instructional expert uses the structure of the domain to sequence the presentation of part-task training. For example, suppose the instructional expert must teach a region of the domain hierarchy as in Figure 7. The instructional expert has determined that it must teach the skills B^1, C^1, and C^2. Our part-task training philosophy dictates that subskills should be trained before the skills they support. The instructional expert then chooses to tutor C^1 and C^2 before tutoring the integration skill B^1. The part tasks are sequenced from the subskills to the parent skills, and so on, up the domain hierarchy.

As we pointed out before, the procedural network and the analysis of the domain expert can only provide information about a subset of elements of the student model. Instruction is a valuable source of diagnostic information about the regions of the student model that are inaccessible to the procedural network/domain expert (as in Figure 8). Part-task training can be designed to elicit information about the inaccessible areas of student model: "Avoid guessing - get the student to tell you what you need to know"[8]. This diagnostic information is all the more



□ = Region accesible to the Domain Expert

▨ = Region accesible to the Instructional Expert

Figure 8. Regions accessible to the Domain Expert's and the Instructional Expert's diagnosis

valuable since it is directly solicited and not deduced with possibly error-prone analysis.

Conclusions

The two expert systems in our ITS use a common representation of the student. The domain expert can observe and understand the student's actions with a procedural network. The procedural network lends itself to an initial classification of observed errors. The classified student errors can then be interpreted for information about specific skills and concepts in the student model. The student model can further refine the possible causes of the student errors. Our ITS exploits the hierarchical structure of the student model for both further diagnosis of the student and remediation of the student. The hierarchical representation of the student model is a sound representation for instruction, specifically part-task training, as well as diagnosis of student deficiencies. Fault isolation algorithms can use the hierarchical student model as a decision tree. The instructional expert uses the hierarchical structure of the student model to control the sequence of training.

References

[1]Spiro, R. J., Vispoel, W., Schmitz, J., Samarapungavan, A., and Boerger, A., *Knowledge Acquisition for Application: Cognitive Flexibility and Transfer in Complex Content Domain,* in *Executive Control Processes* (ed. B. C. Britton), Erlbaum, Hillsdale, New Jersey, 1987, pp. 177 - 199.

[2]Brown, J. S., and Burton, R. R., *A Paradigmatic Example of an Artificially Intelligent Instructional System*, International Journal of Man-Machine Studies, vol. 10, pp. 323 - 339.

[3]Rickel, J., *An Intelligent Tutoring Framework for Task-Oriented Domains*, Proceedings of ITS-88, Montréal, 1988, pp. 109 - 115

[4]Sacerdoti, E. D., *A Structure for Plans and Behavior*, Elsevier-North Holland, New York, 1977

[5]Rickel, J., *ibid.*

[6]Self, J. A., *Bypassing the Intractable Problem of Student Modelling*, Proceedings of ITS-88, Montréal, 1988, pp. 18 - 24.

[7]Frederiksen, J. R., and White, B. Y., *An Approach to Training Based Upon Principled Task Decomposition*, Acta Pyschologica 71 (1989), pp. 89 - 146.

[8]Self, J. A., *ibid.*