A Blackboard-based Dynamic Instructional Planner

William R. Murray

Artificial Intelligence Department
FMC Corporate Technology Center
1205 Coleman Avenue, Box 580
Santa Clara, California 95052
murray@ctc.fmc.com

Abstract[1]

The Blackboard Instructional Planner is a blackboard-based dynamic planner for intelligent tutoring systems. It generates a sequence of lesson plans customized to a student's background, and adaptively replans to handle student requests and unexpected changes to the student model or time remaining. The planner is designed to be generic to tutors that teach troubleshooting for complex physical devices. It controls the Lower Hoist Tutor, a prototype tutor for the Mark-45 naval gun mount. This tutor teaches troubleshooting of the lower hoist, a complex hydraulic-electronic-mechanical assembly of the Mark-45. The tutor implementation demonstrates the planner's operation and means of integration.

This research contributes to an understanding of dynamic instructional planners, planner-controlled tutors, and ITS control architectures. The planner implementation shows precisely how a blackboard architecture can be used to realize a dynamic instructional planner. Although experimental, the tutor implementation demonstrates how such a planner can be embedded in an intelligent tutoring system and what the respective roles of the different components of a planner-controlled tutor are. Finally, the analysis of the planner's use of the blackboard architecture clarifies requirements for control architectures in intelligent tutoring systems and trade-offs made in choosing alternatives.

Introduction

The Blackboard Instructional Planner (BB-IP) is a blackboard-based dynamic planner for intelligent tutoring systems. Although experimental, the planner demonstrates key plan generation and replanning capabilities required to handle common tutorial situations. It generates a sequence of lesson plans customized to a student model inferred from a pre-instruction questionnaire. The content, sequencing, and length of lessons are determined by the inferred student model, by the time allotted for lessons, by the target skill to be taught, and by the subject domain. These lesson plans are revised during instruction in response to student questions and requests, changes in time remaining for lessons, and modifications to the student model.

[1]This project was sponsored jointly by the Manpower, Personnel and Training R&D Committee of the Office of the Chief of Naval Research (under contract N00014-86-C-0487); the Air Force Human Resources Laboratory; the Naval Training Systems Center; and the Naval Personnel Research and Development Center.

The goal of this research is to increase the flexibility and effectiveness of computer-based tutoring systems by applying dynamic planning techniques to control the tutor's actions. Traditional CAI systems do not plan at all although they can deliver very well-crafted lesson plans. Typically, however, they have no domain expertise, customize only via branching or problem selection, and cannot use opportunistic teaching methods. They must limit student questions and requests since they cannot reason about the domain or the student's lesson plan during instruction. On the other hand, most intelligent tutoring systems incorporate domain expertise and student modeling so that they can apply opportunistic teaching approaches in the course of student problem-solving or a dialog about the subject matter. These tutors tend to be used as adjuncts to classroom instruction since they assume primary instruction is delivered elsewhere. Generally, lesson planning is not done although recent systems have focused on sophisticated local discourse planning. The purpose of the Blackboard Instructional Planner is to integrate these two complementary means of instruction - plan-based and opportunistic - for tutors and domains where this is most appropriate.

Our approach to this control problem is called *dynamic instructional planning*. In this approach, an initial instructional plan is generated by the planner. This plan is customized to an inferred student model. It takes into account the resources available to the tutor (e.g., time). The tutor interprets this plan to control its delivery of instruction. The planner is dynamic since it can later revise this plan during instruction as the tutorial situation changes. The tutorial situation changes as the student model changes, the amount of time available changes, and as student-initiated interactions interrupt the tutor's plan.

The planner operates as the control element of an intelligent tutoring system, as shown in Figure 1 (arrows indicate data flow). The planner generates an initial instructional plan customized to the inferred student model. The actions in the instructional plan are procedures that control the text, highlighting, and animation displayed on the student interface. The interface also accepts student input, including student-initiated questions and requests that can interrupt these instructional procedures. The library of possible instructional procedures the planner can draw upon is part of the courseware. The courseware includes curriculum materials such as test questions and troubleshooting cases. The domain expert evaluates student performance on these cases, demonstrates correct troubleshooting, and provides answers to student questions about the domain. These evaluations update the student model. Changes to the student model cause replanning if student progress is much better or worse than expected.

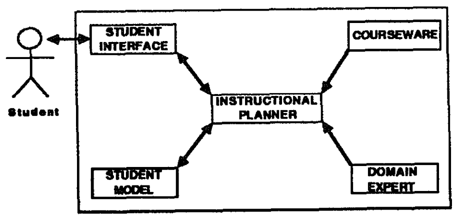

Figure 1: The planner and other ITS components

The planner is designed to be generic to tutors that teach troubleshooting for complex physical devices. It has been implemented as the controller for the Lower Hoist Tutor, a prototype tutor for the Mark-45 naval gun mount, to demonstrate the planner's operation and means of integration. The tutor teaches troubleshooting of the lower hoist assembly, a complex hydraulic-electronic-mechanical assembly of the Mark-45, by first imparting a mental model of the device and its operation. A STEAMER-based (Hollan, Hutchins, & Weitzman 1984) display of the lower hoist schematic has been adapted for use in exposition, assessment, and troubleshooting practice.

Overview of Operation

Figure 2 provides an overview of the operation of the Blackboard Instructional Planner. The *generic skills knowledge base* is a graph with nodes representing device-independent troubleshooting skills for complex hydraulic-electronic-mechanical devices. Arcs represent prerequisite relationships among these skills. The planner interprets this graph as a very high level strategy for teaching troubleshooting.[1] Structure and operation of the device must be taught along with possible faults and a troubleshooting strategy. The *domain knowledge base* is a semantic network that provides the particulars of the structure and operation of the lower hoist assembly of the Mark-45. It includes a parts breakdown and the sequence of events (part state changes) that occur in normal operation. The circled X in Figure 2 means that a cartesian product of the two knowledge bases is formed to produce the graph of *domain-specific skills*. For example, one skill in the generic skills knowledge base specifies that the student should understand the role of each part in the device. So for the lower hoist's 42 parts, 42 domain-specific skills result. Each corresponds to understanding the role of one particular part.

[1]This strategy is not based on any cognitive theory, rather it was inferred from the subject matter expert's approach to classroom instruction.

The graph of domain-specific skills acts as a machine-generated curriculum graph for the skill of troubleshooting the particular device. A three-level instructional plan is produced by selecting out skills from this graph to teach the student, selecting abstract methods to teach the skills, and then selecting specific LISP routines consistent with these methods. The plan generation process also involves sequencing the lessons and lesson plan steps, selecting appropriate parameters for the instructional procedures, critiquing the plan as it is developed, and revising it to improve discourse flow. The result is an *instructional plan*, which is further broken down into a sequence of *lesson plans*, corresponding to individual tutorial sessions.

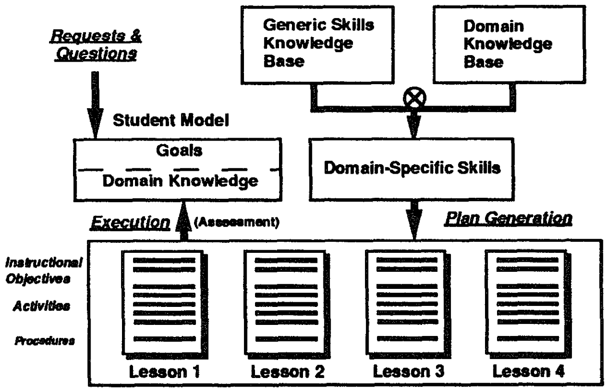

Figure 2: Overview of BB-IP Planning

The plan is executed by interpreting the instructional procedures. Some of the procedures assess student ability through tests or by monitoring task performance. This assessment updates the student model, which can in turn cause replanning to occur. Replanning occurs when student performance is unexpectedly good or bad. It can also be initiated when much less time or more time is available than expected, or when student questions or requests interrupt the tutor's plan. During replanning the plan is edited to add plan steps, remove them, or to alter the procedure parameters. Then execution resumes.

Examples of Plan Customization

Consider two different student backgrounds and the different plans that are generated. Suppose the first student is deficient in the background area of electronics and learns more slowly than the average student. The second student has the expected knowledge of hydraulics, electronics, and mechanics and learns rapidly. The tutor labels the first as *Electronics-Deficient, Low-Aptitude* and the second as *Has-Prerequisites, High-Aptitude* based on the results of a short pre-instruction questionnaire covering prerequisite material and general student background. The inferred cognitive stereotypes (Rich 1979) are used to initialize an overlay student model (Carr & Goldstein 1977) that is an overlay of the domain-specific skills.

The tutor generates quite different plans for these two students although the high-level teaching strategy, derived from the common generic skills knowledge base, is the same. Both plans cover structure and operation of the lower hoist and discuss a general troubleshooting strategy. Then correct troubleshooting is demonstrated and student performance monitored in troubleshooting cases selected by the tutor. But the first plan is much longer than the second since it provides more layered instruction on lower hoist structure and operation, more discussion of how to troubleshoot, and specific remedial instruction covering the operation of electrically-controlled parts.

The instructional plan for the *Electronics-Deficient, Low-Aptitude* student consists of five lessons and 51 steps (instructional procedures). In the first lesson an introduction to lower hoist structure and operation is given, followed by a detailed description of the lower hoist parts, their location, and their role in lower hoist operation. The second lesson explains how solenoids operate and then provides a detailed explanation of all the part state changes that occur in a normal cycle of operation of the lower hoist. This detailed explanation interleaves text explaining part state changes with animation showing the changes discussed. In the third lesson the student must demonstrate his understanding of normal operation by successively pointing to the next part that should change state and indicating what the new state should be. Essentially, the student leads the simulation. In the fourth lesson, the tutor discusses the possible faults that can occur in the lower hoist. It also discusses how to reason from symptoms to possible faults. Then it presents a troubleshooting strategy that takes into account part change cost, malfunction probability, and split-half testing. Finally, troubleshooting is demonstrated for a particular case. The last lesson is devoted to letting the student practice troubleshooting for various cases with the tutor's assistance.

The instructional plan for the *Has-Prerequisites, High-Aptitude* student is quite different. Instead of five lessons and 51 steps there are only three lessons and 26 steps. This plan omits much of the layered instruction given to the first student. For example, the procedures that introduce the structure and operation of the lower hoist in a layered fashion are omitted. Similarly, procedures that introduce troubleshooting in a layered fashion are omitted along with topics that the tutor judges to be of lower priority, such as part locations and descriptions. Procedure parameters also differ for the procedure that monitors student troubleshooting. In the second instructional plan the parameters are set to provide hints later, tolerate more incorrect actions, and provide less assistance with selecting test codes than in the first instructional plan.

Examples of Dynamic Replanning

Consider an example where a question is asked that interrupts the tutor's presentation. Assume the tutor is executing an instructional procedure called *Explain-Subcycles* that explains the detailed sequence of changes that occur during the operation of lower hoist. Midway through the procedure, when a change happens to the UVK4 piston, the student interrupts and asks what the role of that particular part is. The interruption and specification of the question is handled by a menu interface. Question templates are used to restrict questions to those that can be answered by executing one of the tutor's instructional procedures.

The tutor can either defer or directly answer the question. In this case the tutor chooses to answer the question now since the material being asked about will not be covered later. To answer the question the current procedure is suspended and a step to answer the question is inserted. The step *Explain-Subcycles* is suspended and two steps are spliced in after it:

Explain-Subcycles **[subcycles hoist]**
Patch-1:Part-Roles(to-ans-Ques#1) **[UVK4]**
Patch-1:Explain-Subcycles(cont) **[subcycles hoist** ***cont*****]**

The parameters of each procedure are indicated in boldface after the procedure name. They are shown here in a more readable format than the internal representation. The first step after the interrupted *Explain-Subcycles* procedure answers the question. The second continues *Explain-Subcycles* where it was interrupted. But this is not all that happens since the student model is affected simply by the student asking the question.

When the student asks a question about material that was covered earlier, the tutor's belief that the student knows that material is diminished. In this case the tutor covered the roles of the lower hoist parts earlier. The tutor detects that its prior goal, that the student knows the roles of these parts, is no longer satisfied and still should be. To reachieve this goal it splices in a review of the roles of the lower hoist parts before resuming *Explain-Subcycles*. Now the affected portion of the plan, after both plan patches have been added, appears as:

Explain-Subcycles **[subcycles hoist]**
Patch-1:Part-Roles(to-ans-Ques#1) **[UVK4]**
Patch-2:Part-Roles(to-review) **[all hoist parts]**
Patch-2:Multiple-Choice-Quiz **[role each hoist part]**
Patch-1:Explain-Subcycles(cont) **[subcycles hoist** ***cont*****]**

The multiple choice quiz has been added to check that the tutor's review is successful.

Now plan critics are applied to improve the tutorial discourse. There is an abrupt context shift from the review of the roles of the lower hoist parts back to the detailed discussion of the lower hoist cycle. This shift in discourse is smoothed out by adding a step: *Patch-2:Transition* **[part roles, subcycles hoist]**. Another plan critic attempts to remove redundant discussions. It ensures that only the first *Part-Roles* procedure discusses UVK4 by removing UVK4 from the list of parts discussed by the second *Part-Roles* procedure. After all the edits, the affected portion of the lesson plan appears as shown below:

Explain-Subcycles **[subcycles hoist]**
Patch-1:Part-Roles(to-ans-Ques#1) **[UVK4]**

Patch-2:Part-Roles(to-review) **[all parts except UVK4]**
Patch-2:Multiple-Choice-Quiz **[role each hoist part]**
Patch-2:Transition **[part roles, subcycles hoist]**
Patch-1:Explain-Subcycles (cont) **[subcycles hoist *cont*]**

Requests are handled similarly. New steps are inserted to handle requests. Furthermore, future lesson plan steps may be *omitted.* For example, suppose the tutor grants a student request to use the device simulation. The tutor monitors the student's actions. If it believes the student has learned material it plans to cover then it will omit pending procedures to cover this material.

Dynamic replanning can also be initiated by unexpected changes to the student model. For example, as the tutor monitors student troubleshooting performance it updates the student model. Each troubleshooting action the student selects is compared to an expert's to compute its utility. Repeated inappropriate actions and hint requests cause the tutor to revise its belief that the student can troubleshoot the device downwards. When the belief value falls below a certain threshold, the procedure for monitoring student troubleshooting is suspended and replanning begins.

A diagnosis phase is entered where the tutor decides why its instruction has been ineffective. It could be that it assumed too much of the student, or that an earlier method of instruction was inappropriate. Once it has decided the cause of the problem it edits the instructional plan to address the problem and continues. Only a few diagnosis methods have been implemented. One is to collect the prerequisites for the skill not learnt and then to assess the student's knowledge of the prerequisites, one by one, until a prerequisite is found that the student does not understand. Then a review for that prerequisite is spliced into the plan prior to the suspended procedure for monitoring student troubleshooting. Execution resumes with the review before troubleshooting practice continues.

Other kinds of planning functionality have also been implemented, as discussed in detail in (Murray 1990b). If there is insufficient time to finish all the activities of a lesson then some activities may be deferred until the next lesson. All the remaining lesson boundaries may also be affected. If there is extra time a useful activity is added to take advantage of the time remaining. Incremental planning has also been demonstrated. Only abstract activities are specified for lesson plan steps with the choice of procedure deferred until the step is reached.

In the implementation of the examples above specific decisions had to be made about how to customize plans, how to handle student questions, and how to diagnose and correct ineffective plans. The claims made in this paper are not about these specific decisions, since at present there is little guidance from the educational psychology literature specific enough to be applied in these situations. Other approaches can be implemented that would customize plans differently and make different decisions about when and exactly how to replan. The claim made is that the planner framework - the plan representation, the use of blackboard architecture, and the kinds of plan editing that can be performed - would support these alternative pedagogical approaches as well.

Plan Representation

This section presents the hierarchical instructional plan representation. There are three levels:

1. *Instructional objectives* - goals of the tutor. These are domain specific skills which the tutor intends for the student to acquire (e.g., **can-predict-normal-behavior-hoist**).
2. *Activities* - abstract methods for achieving these goals. The most common activities are to cover a topic (e.g., **cover-topic-part-role-uvk4**) or to perform assessment for a skill (e.g., **assess-understands-structure-hoist**).
3. *Procedures* - instructional routines that implement specific activities.

Multiple procedures are available for the same activity. For example, different kinds of tests can be used for student assessment.

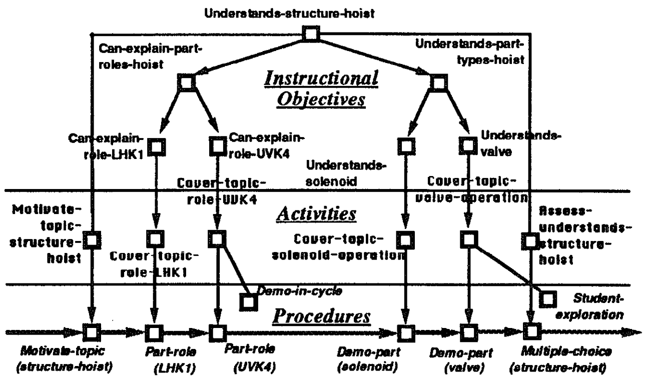
Understands-structure-hoist
Can-explain-part-roles-hoist
Understands-part-types-hoist
Instructional Objectives
Can-explain-role-LHK1
Can-explain-role-UVK4
Understands-solenoid
Understands-valve
Cover-topic-role-UVK4
Cover-topic-valve-operation
Activities
Motivate-topic-structure-hoist
Cover-topic-role-LHK1
Cover-topic-solenoid-operation
Assess-understands-structure-hoist
Demo-in-cycle
Student-exploration
Procedures
Motivate-topic (structure-hoist)
Part-role (LHK1)
Part-role (UVK4)
Demo-part (solenoid)
Demo-part (valve)
Multiple-choice (structure-hoist)

Figure 3: A simple instructional plan

Figure 3 shows a small instructional plan for teaching the structure of the lower hoist, assuming that it only had two parts. The top level is the *instructional objectives level.* On it, the top-level instructional objective has been broken down into two subordinate (i.e., prerequisite) objectives. First, that the student can explain the roles of the parts of the lower hoist. Second, that the student understands how the different part types operate. These objectives are broken down still further. First, the student must understand the role of each of the two parts. Then since LHK1 is a solenoid assembly the student needs to understand how solenoids operate. Similarly, since UVK4 is a hydraulic valve the student needs to understand how hydraulic valves operate.

The reason that the instructional objectives level is important is that it supports replanning. BB-IP needs to know why it was doing a procedure to determine what to do if that procedure fails. This representation of plan rationale supports replanning when previously satisfied instructional objectives are no longer maintained, or when pending instructional objectives are discovered to be already satisfied.

The next level, the *activities level*, supports incremental planning by allowing methods for carrying out the objectives to be represented, without committing to specific actions to carry out these methods. There is one activity for each terminal skill node plus additional activities to improve discourse flow and monitor plan progress. The activities **cover-topic-role-lhk1** and **cover-topic-role-uvk4** simply present text describing the role of the two lower hoist parts. The activity **motivate-topic-structure-hoist** has been added to improve discourse flow. It explains to the student that it is important to understand lower hoist structure in order to perform effective troubleshooting. The activities **cover-topic-solenoid-operation** and **cover-topic-valve-operation** present material explaining the operation of solenoids and valves. The final activity **assess-understands-structure-hoist** tests the student to determine if the plan has had its intended effect.

The final level, representing a sequence of directly executable plan actions, is the *procedures level*. For each activity a single procedure has been chosen to carry it out. For example, to cover UVK4's role the tutor can use either *Part-Role* or *Demo-in-Cycle*. *Part-Role* presents a textual description of UVK4's role. It also highlights and labels it in the device simulation. *Demo-in-Cycle* demonstrates the part's operation in the lower hoist cycle by animating the STEAMER device simulation. Similarly, the topic of hydraulic valve operation can be taught either by *Student-Exploration* or by *Demo-Part*. The first procedure asks the student to use the device simulation to explore the operation of a particular valve. The second demonstrates different valve states and how they affect connecting parts.

Each procedure is broken down into steps called *procedure steps*. A procedure can be interrupted between procedure steps and then resumed later. Typically, a procedure step displays another paragraph of text, or makes an incremental change to the device simulation shown on the student interface. For assessment procedures, each procedure step asks a question. Procedures are written this way so the student can ask questions or make requests between procedure steps. Another reason is to monitor procedure execution and student progress since the tutor can assess progress or lack of progress between procedure steps. If progress is insufficient then the tutor can interrupt the procedure to adjust its parameters or abandon it altogether.

Plan Generation, Execution, and Repair

Figure 4 shows the resources used in plan generation. The *activities library* is a set of generic activities (e.g., **cover-topic, assess-topic, demo-skill)** that can be used to achieve objectives. The *procedures library* is a set of instructional routines that can be used to achieve activities (e.g., *Explain-Subcycles*). These libraries are part of the courseware for the domain, although most of the activities and procedures can apply to other similar domains. The student model is a resource since it is used to filter out those objectives, activities, and procedures that are inappropriate for the student. Another resource is the library of *discourse critics* that are used to critique and revise the plan during plan generation, and immediately after plan editing. Once the plan has been generated, time estimates and a simple hill-climbing procedure are used to place lesson partitions (see (Murray 1990b) for details).

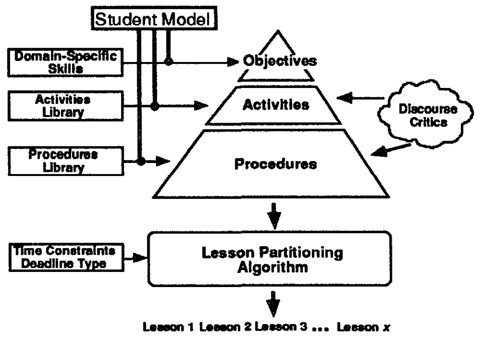

Figure 4: Plan generation

Plan generation and execution occur in phases called *control phases*. These are states where only certain kinds of planning actions are performed. BB-IP's control phases are shown in Figure 5. Phases on horizontal lines are next to the planning level they affect. Planning actions that occur in these phases either add to or modify a partially refined instructional plan. The special *Meta-Level* phase decides what plan phase to jump to next.

Consider plan generation first, assuming complete top-down plan elaboration. The basic sequence of phases is:

1. *Refine Objectives*- The top-level instructional objective is refined by copying prerequisite objectives from the domain specific skills.
2. *Assess Objectives* - The student is given a pre-instruction questionnaire to initialize the student model.
3. *Propose Activities* - Activities to achieve the objectives are proposed.
4. *Prioritize Activities* - Numerical priorities are assigned to each activity to indicate its importance.
5. *Filter Activities* - Only those activities above a threshold are retained.
6. *Sequence Activities* - These activities are now sequenced.
7. *Critique Activities* - Improvements in the discourse flow are made.
8. *Propose Actions* - Candidate procedures for each activity are proposed.
9. *Select Actions* - Heuristics are used to select the best procedure for each activity. Parameters are set for the procedures.
10. *Sequence Actions* - Procedures are

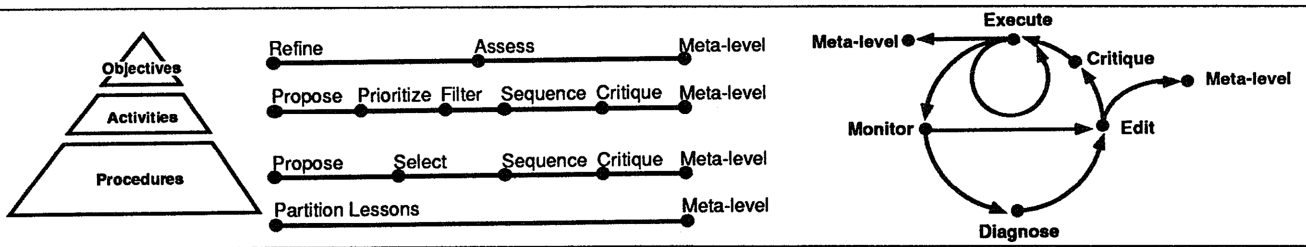

Figure 5: Control phases in plan generation and execution

sequenced in the same order as their parent activities.

11. *Critique Actions* - The plan is again critiqued and improved.
12. *Partition Lessons* - Partitions are laid down, breaking the instructional plan up into lesson plans.

The *Meta-Level* phases at the end of each horizontal line in Figure 5 were not mentioned since they just produce the sequencing above. However, when incremental planning occurs phases 8, 9, 10, and 11 are skipped.

After the *Partition Lessons* phase execution begins. The *Meta-Level* phase moves the planner into the execution loop shown at the right of Figure 5, starting with the *Execute* phase. The planner stays in the *Execute* phase as long as the instructional plan is executing smoothly. Interruptions occur when the student model or time do not change as expected or if there are student-initiated questions or requests. The planner enters the *Monitor* phase and the reason for the plan's interruption is recorded.

A question or a request is deferred if it will be handled later in the instructional plan anyway. Otherwise it is granted immediately. In the former case the tutor explains why the request is being deferred and continues with the *Execute* phase. In the latter case a step performing the procedure that answers the question or grants the request is spliced into the lesson plan in the *Edit* phase. The *Critique* phase adds a transition step then execution resumes. The next step to be executed is the step satisfying the student's request.

An unexpected change to time remaining or the student model means that the plan has some flaw. The *Diagnose* phase attempts to infer what is wrong. This phase ends when a decision has been made about what the problem is. Then the *Edit* phase either adds or deletes lesson plan steps, or alters procedure parameters, according to the kind of problem diagnosed. The steps added may only be at the activity level, requiring a digression back to the plan generation process to select procedures. Then this new plan patch is critiqued and execution resumes.

Planning and execution are interleaved through the use of control phases and in particular the *Meta-level* control phase. The *Meta-level* phase allows suspending the execution cycle to return to any point in the plan generation process. It also determines when execution should resume. This kind of interleaving of control phases occurs in incremental planning, in diagnosing plan execution problems, in generating initial assessment plans, and in elaborating, splicing in, and then executing plan repair patches.

The Blackboard Architecture and its Role

BB-IP is built on the BB1 Blackboard Architecture (Hayes-Roth 1984), which allows multiple blackboards, rather than just one. These are the main blackboards used by BB-IP:

- *Instructional Plan* - represents the three-level instructional plan.
- *Student Model* - represents the student model.
- *Device* - represents the structure and operation of the device.
- *Planner Control* - represents meta-level decisions such as whether to continue planning or start execution. Problems noted in the plan's execution and decisions about their cause are also recorded.

The *Device* blackboard is not updated during execution; the others are.

The knowledge sources of BB-IP are planning actions. They primarily modify the *Instructional Plan* blackboard. The kinds of actions performed are:

- *Plan generation* - new objects are added to the instructional plan and linked to the objects they refine. These objects represent objectives, activities, or procedures.
- *Plan execution* - a step in a procedure is executed.
- *Plan monitoring* - some problem in plan execution is noted. For example, the time available or student model is not as it should be.
- *Plan diagnosis* - a decision is made regarding a plan failure.
- *Plan revision* - part of the plan is modified. New activities or procedures may be added or others deleted.

The execution cycle of BB-IP is shown in Figure 6. Changes to the blackboards are caused either by the execution of planning actions, or by student interaction. These changes are called *events*. They trigger *knowledge sources* creating activation records which are placed on the agenda. These *knowledge source activation records* or *KSARs* record the knowledge sources and their variable bindings when triggered. The *scheduler* selects the next KSAR to run. Then the next KSAR is interpreted. Its execution may cause changes to the student interface, the instructional plan, or both. This cycle continues until the instructional plan has been completed.

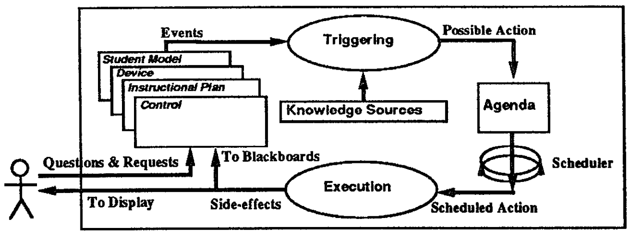

Figure 6: BB-IP Execution Cycle

The blackboard architecture facilitates the implementation of dynamic instructional planners by supporting an explicit plan representation, meta-level reasoning, interruptibility, and rapid prototyping. The explicit instructional plan representation supports dynamic planning by representing plan rationale, future unexecuted actions, global resource allocation, and constraints on partially refined plans. The use of plan critics is also supported. These issues are explained in more detail in (Murray 1990b). Other reasons cited (Hayes-Roth 1987a) for using blackboard architectures such as integrating uncertain data, island driving, and providing opportunistic control appear more relevant to interpretation applications (e.g., HEARSAY-II (Erman et al. 1980)) than this planning application.

Related Work

This section considers related work in control for intelligent tutoring systems. We consider:

1. *Reactive systems having no plans at all.*
2. *Tutors that follow a single fixed plan.*
3. *Tutors that can only select dynamically from stored plans.*
4. *Tutors that generate and interpret an explicit instructional plan.*

Reactive systems recognize and act upon opportunities to provide instruction but do not plan any sequence of actions. WEST (Burton & Brown 1979) and SOPHIE-I (Brown, Burton, & Bell 1975) are examples. Reactive systems such as these are limited to coaching or opportunistic tutoring strategies. The student's actions determine the course of the instructional session, not a stored or generated plan. Simulated microworlds that only allow discovery learning can also be considered reactive systems, but it is arguable whether such systems are intelligent tutoring systems, although they do apply artificial intelligence techniques to education.

Traditional CAI tutors rely on a single fixed plan. The instruction produced may be very good but these systems cannot be readily extended to new domains or instructional strategies since they procedurally encode domain knowledge and teaching methods. To make program development practical (i.e., less time consuming), their flexibility is quite limited. The tutor controls the instructional session, rather than the student. Student questions and requests are restricted or disallowed. For example, the student typically cannot pose a problem for the tutor to solve.

A considerable increase in flexibility results when tutors can dynamically select discourse plans from a plan library. MENO-TUTOR (Woolf & McDonald 1984) provides these capabilities with pre-stored skeletal plans, encoded as default state transitions in an ATN-like discourse management network. The network is augmented with meta-rules that allow transitions from the execution of one skeletal plan to the execution of another. Another tutor, WHY (Stevens & Collins 1977), also provides Socratic question and answer dialogs, but it uses rules to generate its dialogs. GUIDON (Clancey 1987) provides more sophisticated capabilities than either WHY or MENO-TUTOR. Its t-rules (tutorial rules) are used both as pre-stored discourse plans and as meta-rules to control the selection and operation of other t-rules. GUIDON's capabilities are still limited by the lack of an explicit instructional plan even though it provides very sophisticated dialog management. GUIDON's dialog can be verbose since it does not plan its dialog to determine which topics should be stressed and which should be deemphasized. Although the user can interrupt GUIDON at any time some requests may be disallowed since they would invalidate assumptions underlying the t-rule currently executing. These and other limitations are discussed in more detail in (Murray 1990a).

Tutors that generate and revise explicit instructional plans during instruction are intended to overcome these limitations, and support both lesson and discourse planning. This paper has described one approach; others are described briefly below. A planner-based approach to instruction based on applying classical (i.e., STRIPS-based (Fikes & Nilsson 1971)) planning techniques is presented in (Peachey & McCalla 1986). The focus is on planning lesson content. Unlike the Blackboard Instructional Planner, this planner does not address issues in resource management, incremental planning, handling mixed-initiative instruction, and integrating lesson planning and discourse management. Another tutor, IDE-Interpreter (Russell 1988), also adopts a planner-based approach to instruction to interpret an instructional design developed in IDE (Russell, Moran, & Jordan 1988), the Instructional Design Environment. It does not rely on classical planning techniques but uses a top-down plan elaboration approach similar to that presented here. However, it does not address issues in procedure interruption and the diagnosis of failed plans. The handling of mixed initiative instruction is also more limited. Finally, the work presented here builds on research by MacMillan and Sleeman (MacMillan & Sleeman 1987). Their work focused on implementing general planner and ITS architectures. The subsequent work by the author has focused on the implementation of a dynamic instructional planner for a realistic application, rather than developing generic ITS architectures.

The planner presented here is actually the second version of the Blackboard Instructional Planner. The first version was much simpler and focused more on discourse management and less on lesson planning. Instructional actions were simulated and resource management not addressed. It was implemented using a high-level language describing instructional actions. The implementation was modeled after the PROTEAN (Hayes-Roth et al. 1987b) blackboard application and its ACCORD (Hayes-Roth et al. 1987c) language framework using Hayes-Roth's BB* environment as described in (Murray 1989a).

Research Contributions

The first major contribution of this research is its analysis of architectural and planning issues in ITS. (Murray 1990b) presents a more detailed analysis of the role of the blackboard architecture than can be provided here. Alternative approaches to planning for ITS, and trade-offs made between them, are considered in (Murray 1989b). Two particular architectures for intelligent tutoring systems, discourse management networks and blackboard architectures, are analyzed in (Murray 1990a). That paper concludes that blackboard architectures provide greater support for the implementation of dynamic instructional planners, although discourse management networks are sufficient for sophisticated dialog management where only local planning is required.

A second research contribution is the implementation of the Lower Hoist Tutor. This prototype tutor shows how a dynamic instructional planner such as BB-IP can be integrated with other ITS components to form a complete system. A realistic naval training application, teaching troubleshooting of the lower hoist assembly of the Mark-45 naval gun mount, has been selected to test the planner. Although not yet appropriate for classroom use, all instructional actions have been implemented and the tutor can deliver about two hours worth of instruction on the lower hoist.

The major research contribution of this work is the implementation of the Blackboard Instructional Planner. This planner shows precisely how a dynamic instructional planner can be implemented in the blackboard architecture. The Blackboard Instructional Planner generates an instructional plan customized to student background and time constraints, then adaptively replans to support mixed-initiative instruction, and to handle changes to the student model and time remaining.

Acknowledgements. I would like to thank Kurt Steuck (AFHRL) and Susan Chipman (ONR) for their technical advice and guidance. I am also grateful to Perry Thorndyke, Barbara Hayes-Roth, N. S. Sridharan, Stuart MacMillan, and Lee Brownston for their help on this project.

References

Brown, J. S.; Burton, R. R.; and Bell, A. G. 1975. SOPHIE: a Step towards a Reactive Learning Environment. *IJMMS* 7:675 - 696.

Burton, R. R.; and Brown, J. S. 1979. An Investigation of Computer Coaching for Informal Learning Activities. *IJMMS* 11:5 - 24.

Carr, B; and Goldstein, I. P. 1977. Overlays: a Theory of Modeling for Computer-aided Instruction, Technical Report, AI Lab Memo 406, MIT.

Clancey, W. 1987. *Knowledge-based Tutoring: The GUIDON Program.* The MIT Press.

Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; and Reddy, D. R. 1980. The Hearsay-II Speech-understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys* 12:213 - 253.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189-208.

Hayes-Roth, B. 1984. BB1: An Architecture for Blackboard Systems that Control, Explain, and Learn about their own Behavior, Tech Report HPP 84-16, KSL, Stanford.

Hayes-Roth, B. 1987a. Blackboard Systems. In *Encyclopedia of Artificial Intelligence*, 73 - 80. Wiley-Interscience Publication.

Hayes-Roth, B.; Buchanan, B.; Lichtarge, O.; Hewett, M.; Altman, R.; Brinkley, J.; Cornelius, C.; Duncan, B.; and Jardetzky, O. 1987b. PROTEAN: Deriving Protein Structure from Constraints. In *Blackboard Systems*, 417 - 431. Addison-Wesley.

Hayes-Roth, B.; Garvey, A.; Johnson, M. V.; and Hewett, M. 1987c. A Modular and Layered Environment for Reasoning about Action, Technical Report KSL 86-38, KSL, Stanford.

Hollan, J. D.; Hutchins, E. L.; and Weitzman, L. 1984. STEAMER: an interactive inspectable simulation-based training system. *AI Magazine* 5(2):15 - 27.

MacMillan, S. A.; and Sleeman, D. H. 1987. An Architecture for a Self-improving Instructional Planner for Intelligent Tutoring Systems. *Computational Intelligence* 3(1):17 - 27.

Murray, W. R. 1989a. Dynamic Instructional Planning in the BB1 Blackboard Architecture. In *Blackboard Architectures and Applications*, 455 - 480. Academic Press, Inc.

Murray, W. R. 1989b. Control for Intelligent Tutoring Systems: a Blackboard-based Dynamic Instructional Planner. Proceedings 4th International Conference on AI and Education, 150 - 168. IOS.

Murray, W. R. 1990a. Control for Intelligent Tutoring Systems: A Comparison of Blackboard Architectures and Discourse Management Networks. *Machine-Mediated Learning* 3(1), 1990.

Murray, W. R. 1990b. A Blackboard-based Dynamic Instructional Planner. Tech Report R-6376, FMC Corporation. ONR final report.

Peachey, D. R.; and McCalla, G. I. 1986. Using Planning Techniques in Intelligent Tutoring Systems. *IJMMS* 24:77 - 98.

Rich, E. 1979. User Modeling via Stereotypes. *Cog. Sci.* 3:355 - 366.

Russell, D. M. 1988. IDE: The Interpreter. In *Intelligent Tutoring Systems: Lessons Learned*, 323 - 349. Lawrence Erlbaum Associates.

Russell, D. M.; Moran, T. P.; and Jordan, D. S. 1988. The Instructional-Design Environment. In *Intelligent Tutoring Systems: Lessons Learned*, 203 - 228. Lawrence Erlbaum Associates.

Stevens, A. L.; and Collins, A. 1977. The Goal Structure of a Socratic Tutor. Proceedings National ACM Conference, 256 - 263. ACM.

Woolf, B. P.; and McDonald, D. D. 1984. Building a Computer Tutor: Design Issues. *IEEE Computer* 17(9):61 - 73.