

TAP-2: A Framework for an Inquiry Dialogue Based Tutoring System

L.H.Wong, C.Quек, and C.K.Looi, *Intelligent Systems Laboratory, Nanyang Technological University, School of Applied Science, Nanyang Avenue, Singapore 639798.*

Abstract. In the TAP (Tutoring Agenda Planner) project, we study the feasibility of converting the inquiry teaching method of Collins & Stevens (1982a) into a computational model based on the approach of Intelligent Tutoring Systems (ITS). Inquiry teaching is a dialogue-based teaching style that has the objective of training the student in systematic reasoning process. This paper describes the software architecture for TAP-2 - an inquiry tutoring shell based on the theory of inquiry teaching. To complement the “localized” dialogue planning framework inherent in Collins & Stevens’ theory of inquiry teaching, the TAP-2 architecture has adopted the “global” curriculum planning technique proposed by Peachey & McCalla (1986).

INTRODUCTION

The main focus of the TAP (Tutoring Agenda Planner) project is to study the feasibility of implementing the theory of inquiry teaching by Collins and Stevens as an Intelligent Tutoring System (ITS). It seeks to construct a shell for inquiry teaching software by transferring Collins & Stevens’ (1982a) cognitive theory for inquiry teaching into a computational model.

The research effort consists of two phases. The first phase of the project has been completed in January 95 and has produced the TAP-1 architecture (Wong, 1998; Wong, Quек & Looi, 1998), an ITS shell for delivering minimal inquiry teaching sessions. TAP-1 also adopted Peachey & McCalla’s (1986) “global” curriculum planning technique to complement Collins & Stevens’ theory that concentrates on the “local” delivery planning of inquiry teaching. The integrated planner is supported by an executor with various teaching strategies and a dialogue manager.

This paper focuses on the second phase of this project that embarks on the design and implementation of the TAP-2 architecture. TAP-2 is a more “sophisticated” inquiry teaching shell that is able to support an advanced level of inquiry teaching. In addition, it provides the mechanism for switching between inquiry and other supplementary teaching styles within an instructional session, since inquiry teaching itself is not necessarily suitable for teaching all kinds of knowledge. This tallies with Elsom-Cook’s (1988) argument that using multiple pedagogic strategies (teaching styles) can provide a very powerful learning environment. Hence, as compared against TAP-1, which is a simple inquiry planner, TAP-2 could serve as a basis for developing more complete domains. A software tutor, PADI-2, which teaches rice growing factors (geography), has been developed to demonstrate the capability of TAP-2.

INTELLIGENT TUTORING SYSTEM

ITSs are computer programs that use AI techniques to help a person learn. The basic structure of an ITS is often described by modules representing knowledge of four areas (Hietala, 1989) as depicted in Figure 1. The *domain expert module* (also commonly known as *domain module* or *domain knowledge base*) contains the knowledge of the subject matter to be taught. The *tutoring module* (or *instructional module*; *pedagogic module*) represents the rules, strategies and processes involved in the way the system communicates with the student (i.e., instructional

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

planning). The *student-model module* (or *student modeller*) contains the knowledge (*student model*) and is accessed by the system to ascertain the thinking and strategies of the individual student on the specific subject matter that he is learning. The *communication module* provides the actual interface to the student.

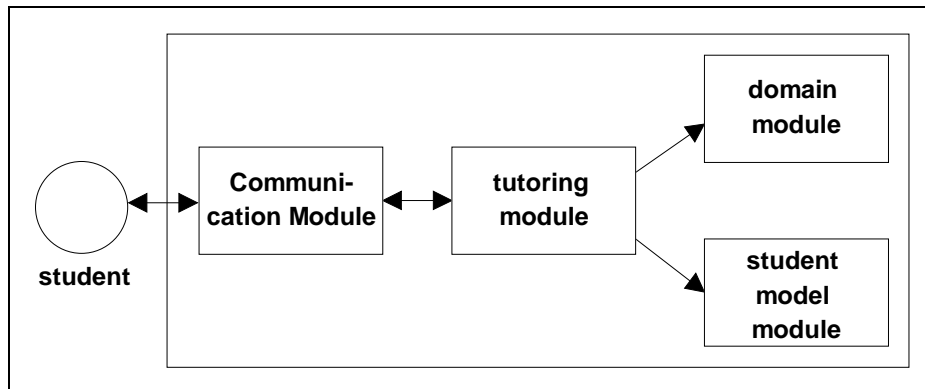


Figure 1. Architecture of typical ITS

Empirical studies of classroom instructors (Peterson et. al., 1978; Zahorik, 1975) have demonstrated that human teachers plan their instructions in a hierarchical fashion. With reference to their findings, recent literature of tutoring module (e.g., Peachey, 1983; MacMillan et. al, 1986; Ohlsson, 1986; Derry et. al., 1988; de Castro et. al. 1988; Murray 1989) stresses the need of separating the planning process into “globalized” curriculum planning and “localized” delivery planning, although the detailed planning policies and approaches varied from work to work. Curriculum planning (or content planning) involves selecting the content for an instructional goal (i.e., the piece of knowledge to be taught) that places a student on an appropriate learning path and planning to achieve that goal. Following the identification of the content, delivery planning determines the most appropriate delivery method in which to carry out the curriculum plan (Wasson 1990).

INQUIRY TEACHING

Inquiry-oriented instruction has been characterized in a variety of ways over the years (Collins, 1982a; DeBoer, 1991; Rakow, 1986) and promoted from a variety of perspectives. In this work, we adopted the characterization of Collins that is probably the best specified.

Inquiry teaching is a dialogue-based teaching style. It forces students to actively engage in articulating theories and principles that are critical for an in-depth understanding of a domain. The knowledge that can be acquired is not simply content, but also the methodology that can be employed in problem solving and the making of predictions. Students are required to formulate hypotheses; for example: to discover generalities based on observation of varied cases - an inductive learning process, so as to force a greater depth in his processing of new information. In other words, it trains the students in systematic reasoning skills.

Figure 2 shows the general steps involved during the systematic reasoning process. The core of the process is the loop consisting of “forming/debugging hypothesis” and “testing hypothesis”. Once a hypothesis is successfully verified, it becomes a rule or theory. One then exits from the loop and proceeds to “make predictions in novel cases” and to “apply knowledge to benefit the real life”.

A dissection of dialogue between PADI-2 and a real student is shown in Figure 3. This serves as an illustration of an inquiry dialogue. In the subsequent sections, we will refer to the dialogue and explain the symbols and strategies used in the brackets of Figure 3.

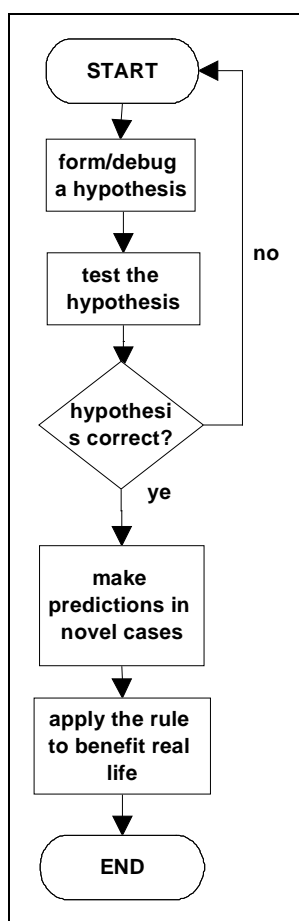


Figure 2. Basic systematic reasoning process

(T = Tutor; S = Student)
 (Session 1: Goal - "Water is necessary for rice growing"; Subgoal - s_for)
 (Student Model: blank)

01 T: Consider Yangtze Plain and Nigeria. Do you know why Yangtze Plain produces rice while Nigeria does not? [CSS6: Differentiation exemplar of factors (minimal pair)]
 02 S: Because Nigeria is short of fertile soil.
 03 T: You think fertile soil is necessary for rice growing. [ES2: Ask if a rule is correct or incorrect]
 04 S: Yes.
 05 T: You are right. Fertile soil is necessary for rice growing. However, Nigeria actually has fertile soil. [IS7c: Suggest values of factors] Please think about other possibility. [IS3: Ask for formulation of an alternative rule]
 (Student Model: add - understood "fertile soil as a necessary factor")
 06 S: Because Nigeria lacks rainfall.
 07 T: Do you think heavy rainfall is a necessary factor for rice growing? [ENS2: Rule based on unnecessary factors] [ES2: (see above)] [ES5: Ask if factors are necessary or unnecessary]
 08 S: Yes.
 (Student Model: add - bug "rainfall as a necessary factor")
 (Note: Water can be obtained from rainfall OR irrigation, so rainfall is NOT necessary for rice growing.)

(Session 2: Goal - same with Session 1; Change subgoal to: s_deb)

09 T: Okay. You have made a guess. Let us test it to see whether it is correct. Now, back to the case of Yangtze Plain. Suppose Yangtze Plain doesn't have heavy rainfall, do you think it still grows rice? [CSS14: Construct a hypothetical case for unnecessary factors]
 10 S: Yes.
 11 T: Why? [IS6: Ask for relevant factors]
 12 S: Because Yangtze Plain has irrigation facilities.
 13 T: So you think either heavy rainfall or irrigation facilities supplies water for rice. [ES2: (see above)]
 14 S: Yes.
 (Student Model: remove - bug "rainfall as a necessary factor"; add - understood "water supply as a necessary factor")
 15 T: Bravo. Water supply, which can either be obtained from heavy rainfall or irrigation, is necessary for rice growing. (proceed to teach other goals)

Figure 3. A dissection of dialogue between PADI-2 and a student

COLLINS & STEVENS' THEORY FOR INQUIRY TEACHING

To formalize the inquiry teaching style, a cognitive theory was proposed by Collins & Stevens (1982a). This theory has been developed almost inductively by observing human teachers (Collins, 1988). It consists of three parts: (1) the goals and subgoals of teachers; (2) the strategies used to realize different goals and subgoals; and (3) the control structure for selecting and pursuing different goals and subgoals.

Teachers typically pursue several subgoals simultaneously. Each goal is associated with a set of strategies for selecting cases, asking questions, and giving comments. In pursuing goals simultaneously, teachers maintain an agenda that allows them to efficiently allocate their time among the various goals (Collins et al., 1975; Collins, 1977).

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

There are two top-level goals that inquiry teachers pursue: (a) teaching students particular rules or theories; (b) teaching students how to derive rules or theories. There are several subgoals with each of these top-level goals. The goals and subgoals that have been identified are shown below. The listed subgoals are similar to the steps involved in the systematic reasoning process shown in Figure 2.

1. Teach a general rule or theory
 - a) Debug incorrect hypothesis, and
 - b) Teach how to make predictions in novel cases.
2. Teach how to derive a general rule or theory
 - a) Teach what question to ask,
 - b) Teach how to formulate a rule or theory,
 - c) Teach what is the nature of a rule or theory;
 - d) Teach how to test a rule or theory, and
 - e) Teach students to verbalize and defend rules or theories.

Furthermore, Collins & Stevens (1982a) have decided to focus on 10 of the most important strategies that inquiry teachers use. These strategies are adopted by TAP-1. Nevertheless, in a later paper (Collins & Stevens, 1982b), they attempted to analyze the strategies of the “very best teachers” for whom they could obtain films or transcripts. Looking at the fine structure of the dialogues, Collins & Stevens noted recurring patterns of strategies in selecting cases, asking questions, and making comments. They characterized the individual strategies in terms of condition-action pairs or productions. These rules can be taken as the finer-grained version of the above-mentioned 10 strategies. These are classified into four following categories; namely,

1. Case Selection Strategies (16 productions, denoted as CSS)
2. Entrapment Strategies (12 productions, denoted as ENS)
3. Hypothesis Identification Strategies (17 productions, denoted as IS)
4. Hypothesis Evaluation Strategies (14 productions, denoted as ES)

Two of the inquiry strategies that are used in the dialogue are elaborated in Figure 4.

CSS6: Differentiation exemplar for factors (Minimal Pair)

- IF (1) a student has not identified one or more factors that are relevant to a particular value on the dependent variable, (*for sentence 01 in Figure 1: relevant factor - water supply; dependent variable - rice growing*) AND (2) there is a case identified that is a positive or negative exemplar of those factors,
(*for sentence 01: case I - Yangtze Plain -- as a positive exemplar*)
- THEN (3) pick a case that has a different value from the previous case on the given factor, that has the same or similar values on other factors, and that has a different value on the dependent variable.
(*for sentence 01: case II - Nigeria -- as a negative exemplar*)

IS6: Ask for relevant factors

- IF (1) there are either necessary or sufficient factors that have not been identified,
(*for sentence 11: also refer to 09 and 10: unidentified factor is irrigation*)
- THEN (2) ask the student for any relevant factors.
(*for sentence 11: “Why?”*)
-

Figure 4. Condition-action pairs of CSS6 and IS6

The control structure that the teacher uses to sequence different goals and subgoals consists of four basic parts: (a) a set of strategies for selecting cases; (b) a student model; (c) an agenda; and (d) a set of priority rules for adding goals and subgoals to the agenda.

Given a set of top-level goals, the teacher selects cases that optimize the ability of the student to master those goals. Subsequently, the teachers begin by questioning the student about the cases and the rules interrelating them. The answers reveal what the student does and does not know. This will modify the student model. Whenever specific bugs in the student's theory or reasoning processes are identified subgoals are created to correct the bug.

An important observation made by Collins from his inspection of inquiry dialogue is that good inquiry teachers tend to stick to a case and make sure that they have used it in all possible ways to deliver the dialogue before they switch to another case. "Drill a case (that the student is familiar with) to death" will free the student from finding out the unfamiliar background in new cases. Hence, he will be able to focus his attention on the conceptual knowledge and the systematic reasoning skill. This aspect of inquiry teaching has not been mentioned in any of Collins or Stevens' papers (Collins, 1995).

Among others, one unique feature of this theory is that it is intended more to develop higher level thought processes than to develop domain-specific knowledge; that is, the reasoning skills that scientists need arise from this model of teaching (Collins, 1987). Hence, Collins & Stevens' theory is best suited for domain that involves rule-based conceptual knowledge, such as science, physical geography, mathematics, moral education, etc. However, it is less useful for domains that teach simple facts, procedural knowledge or skills.

However, the cognitive theory does not address the issue of the ordering of the top-level teaching goals. It concentrates on "local" or "intra-goal" planning but does not include planning at the global curriculum level. In this respect, the theory is incomplete and hence has to be supplemented by other "global" planning techniques; for example, the architecture proposed by Peachey and McCalla (1986).

PEACHEY & MCCALLA'S CURRICULUM PLANNER

Peachey & McCalla (1986) noted that an underlying weakness of ITS is the absence of "global" knowledge about the course being taught. Domain knowledge is stored in the form of bits and pieces. Hence, they proposed that planning techniques be used to create large, individualized courses that could handle the broader subject areas.

The architecture consists of five components: a domain knowledge database, a student model, a collection of teaching operators, a planner and a plan executor. The planner develops a teaching plan that is tailored to a particular student being taught. The executor then uses the teaching plan to guide the student through the course.

Figure 5 depicts a simple curriculum network to teach Ohm's Law. A curriculum network consists of pre-conditions, teaching operators and expected effects. Ohm's Law states that $Voltage(V) = Current(I) * Resistance(R)$. The student who is taught the law should have pre-requisite knowledge of V, I and R. Hence the three concepts are identified as the pre-conditions. The teaching operator defines the teaching action. Ohm's Law is the expected effect of the teaching operator. The network could be expanded to produce a curriculum network for linear circuit domain; for example, adding in the teaching operator for Kirchhoff's Law where Ohm's Law and some other relevant concepts are its pre-conditions.

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

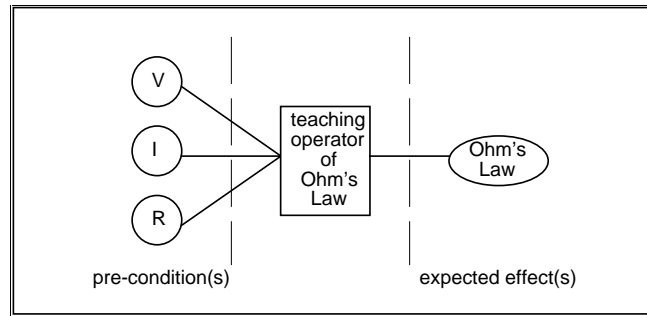


Figure 5. A simple course network teaching Ohm's Law

STRIPS-like planning technique is used to plan the global strategies using the local information: teaching operators (Fikes & Nilsson, 1971). The teaching operators are stored independently in the domain knowledge base. At the beginning, the planner retrieves the teaching operators and links them together, according to the pre-requisite (pre-condition) relationships, to form a curriculum network (a directed graph). Next, the planner attempts to find a path in the network that will lead to the accomplishment of the “ultimate” goal(s). This represents the eventual concepts or skills to be taught.

The executor starts teaching the sequence of actions and diagnosing the student's answers. If a teaching action fails to make the student understand the expected effect(s), the selected path is broken and a new path has to be identified. Finally, if all possible paths are broken, the student is considered to have failed to acquire the ultimate knowledge of the course.

TAP-2 ARCHITECTURE

The most significant challenge in developing TAP-2 is the formalization of the inquiry planning mechanism. Although Collins & Stevens' theory has convincingly been developed by generalizing their empirical observations of human teachers' delivery of inquiry instructions, many detailed or minor planning policies that are needed in each planning step have not been addressed in the relevant literature.

To remedy this deficiency, we have specified extra “non-inquiry” planning rules (see later sessions) which are meant for feeding into various planning modules in order to “tidy up” the architecture of TAP-2. Most of these planning rules are somewhat ad-hoc since they are basically our hypothetical views of instructional planning. Formalization of these ad-hoc rules, for instance, by observing human teachers' instructions, is not recommended because (1) it is out of the scope of the project; (2) it seems redundant as such work is not new to education and the ITS communities. Future work should be done to solve this problem by studying relevant literature and integrating other suitable ideas into TAP-2.

Figure 6 depicts the software architecture of TAP-2. A complete TAP-2-based ITS consists of two major parts: the TAP kernel and the domain-dependent module. Currently, a text-based prototype of TAP-2 (written in C) is currently available on both a Sun™ workstation (4.1 UNIX™) version and an IBM®-compatible PC (MS®-DOS™ and Windows95™) version. A Java Applet™ version with a GUI has also been developed and ported to the World-Wide Web (<http://uranus.sas.ntu.ac.sg:8000/>).

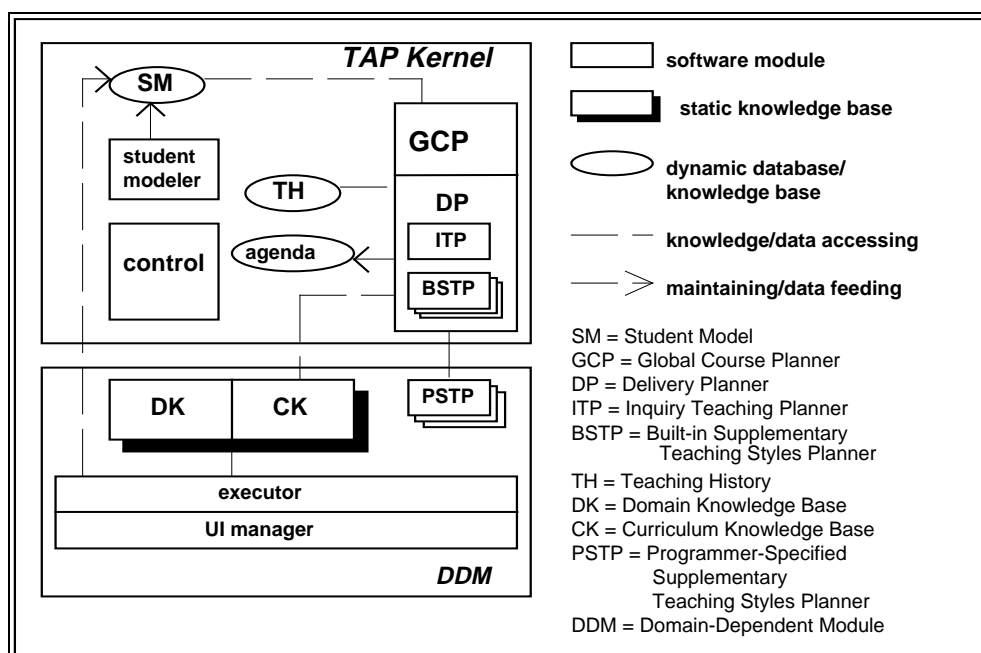


Figure 6. System architecture of TAP-2

The TAP Kernel

The TAP kernel is essentially a domain-independent ITS shell, specifically for “inquiry-centered” dialogue. It consists of three software modules: a control module, a planner and a student modeler; and maintains three databases: an agenda, a tutoring history (TH) and a student model (SM).

In particular, the planner combines Peachey & McCalla’s global curriculum planner (GCP) and a delivery planner (DP). It retrieves the top-level goals specified by the domain and hence produces and maintains the agenda. The DP is further sub-divided into several sub-planners to facilitate switching between different teaching environments (they are instances of Built-in Supplementary Teaching Styles Planner or BSTP). Currently, there are only three teaching sub-planners specified in the DP: (1) Inquiry Teaching (Sub-) Planner (ITP); (2) assessment sub-planner; and (3) expository sub-planner. The agenda stores the goals to be pursued in certain order. The SM, maintained by the student modeler, reflects what the student understands at any time.

The domain-dependent module (DDM) is to be developed by “domain programmers” and subsequently integrated with the TAP kernel. The domain knowledge base in TAP-1 is broken down to two interrelating knowledge bases: (1) a domain knowledge base (DK) that stores the knowledge of the subject domain itself; (2) a curriculum knowledge base (CK) that provides course information and course specifications needed for various steps of the planning process (e.g., local information for GCP, ultimate goal specifications, pre-set SM etc.). Other software modules in the DDM are an executor and a user-interface manager. The domain programmer could also develop a “programmer-specified supplementary teaching styles planner” to introduce more teaching environments to the system.

Subgoals in TAP-2

Collins & Stevens, in various papers (1982a; 1982b; Collins, 1988), have identified some subgoals for inquiry teaching. Each of these subgoals corresponds to a systematic reasoning step. We have re-organized them and added in new delivery subgoals that are not directly

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

concerned with inquiry teaching to provide consistency, coherence, and continuity of instruction. The subgoals are listed in Figure 7.

More subgoals may be added when additional supplementary teaching styles are incorporated in the system. Furthermore, for a given teaching goal, it is unnecessary to go through all the listed subgoals. The domain programmers are responsible for determining the combination of the subgoals needed for every goal. For example, it is not worthwhile to create a lot of subgoals to deliver facts, and concepts in smaller scale.

notation s	subgoals	remarks	sub-planner
s_smrc	recap/summary by presentation	beginning or end of the goal	expository
s_int	Introduction to the course	beginning of goal	expository
s_pre	pure presentation	at any suitable time	expository
s_fin	find out what the student knows	an inquiry subgoal	inquiry
s_for	teach how to form a hypothesis	an inquiry subgoal	inquiry
s_deb	debug an incorrect hypothesis	an inquiry subgoal	inquiry
s_nat	teach nature of a rule	an inquiry subgoal	inquiry
s_tes	teach how to test a hypothesis	an inquiry subgoal	inquiry
s_def	teach how to verbalize/defend a rule	an inquiry subgoal	inquiry
s_nov	teach how to make novel predictions	an inquiry subgoal	inquiry
s_app	teach how to apply a rule	at inquiry subgoal	inquiry
s_inb	inform the student his bug	if bug cannot be removed	expository
s_ase	assess the student	at any suitable time	assessment
s_rem	remove bugs other than wrong hypothesis	-	inquiry or other teaching style
s_com	present briefly forthcoming topic(s)	end of the goal	expository

Figure 7 Subgoals in TAP-2

The Planner

The planner resides in the TAP kernel. It combines the global planning capability of GCP and local planning capability of DP. Computationally, the design of the planner is an attempt to integrate the state-space (GCP) and rule-based (ITP) planning approaches. The planner continuously decides the next goal + subgoal + strategy (+ case, if needed) combination to be executed by referring to the SM and the domain knowledge. The curriculum knowledge base (CK) provides the curriculum specifications and course information needed for various steps of the planning process in both GCP and ITP respectively.

GCP: Global Curriculum Planner

The design of GCP in TAP-2 basically follows Peachey & McCalla's work. The whole planning process in GCP is summarized in the following steps:

- (1) selects a goal from the list of ultimate goals (specified in CK);

- (2) finds the shortest unblocked path leading to the selected ultimate goal by employing the breadth-first search fashion (linking up relevant “precondition-expected_effect” goal pairs, that is, the local information);
 - (3) forms the list of eligible goals and prioritizes them using the preference rating information associated to each goal (specified in CK);
 - (4) removes the first goal from the eligible list and sends it to DP for further planning.
- There are two significant modifications against the original framework which are described below.

(1) Prerequisite of Goals

In Peachey & McCalla’s planner, failure to achieve a pre-condition (a goal) makes its subsequent teaching operators along that path where the pre-condition resides unexecutable (that is, the path is blocked). A goal is simply a requirement of teaching a fact or the nature of the concept (s_nat in the context of TAP-2).

In TAP-2, however, the term “goal” is defined differently as it contains several subgoals to be achieved, probably including, s_for, s_nat, s_tes, s_nov and s_app. If either one of these subgoals is failed, the goal itself is considered unachieved, in terms of inquiry teaching. However, as long as the student has already understood the nature of the concept (that is, achieved s_nat), the system could still proceed to pursue the next operator which has that goal as its prerequisite, even if the rest of the subgoals for that goal are unachieved.

Hence, the prerequisites of the teaching operators in TAP-2 are expressed in term of goal + subgoal instead of overall goals. This is to reduce the possibility of blocked learning path and allows the instruction to be carried out smoothly, as long as the student can continue. However, the ultimate goals should be achieved ‘thoroughly’ (that is, all the subgoals associated with them are completely achieved), otherwise this defeats the purpose of inquiry teaching.

(2) Level- and Discipline-Independent Domain

In real life, a particular domain may be taught at different levels in a school (with varying difficulties), or different disciplines of a college (with varying emphasis). We propose a mechanism to handle the curricula of multiple levels/disciplines in a “single-domain” ITS. Large-scale DK and CK that contain all the teaching goals at various levels/disciplines are developed, which facilitate domain programmers to design multi-level/discipline curricula. For each level/discipline, domain programmers are required to specify (1) the pre-set SM (that is, the prerequisites of the level/discipline); and (2) the ultimate goal(s). When the ITS is invoked, the user will be asked to select the desired level/discipline, which in turn loads the corresponding pre-set SM and ultimate goals into the active SM and the GCP respectively. Such a mechanism could even be applied to break down the curriculum within a level/discipline into several portions or chapters for separate learning hours.

There are two advantages of implementing such a mechanism. Firstly, the redundancy of developing several ITS of interrelated domain knowledge is avoided, since one ITS is reusable for various curricula. The effort saved is particularly significant for developing ITSs with complex domains such as inquiry teaching systems. Secondly, it is easier to permit students who do not really understand or have forgotten the prerequisites of the level/discipline (they have learnt them in earlier chapters or earlier years) to revise them, since the knowledge representations of the prerequisites are already in the DK. Of course, there are trade-off between a huge domain and the availability of storage space, memory space, maintainability and execution time.

DP: Delivery Planner

The DP is given a current eligible goal by GCP. On the basis of this information, DP passes the combination of goal + subgoal + strategy (+ cases, if any) as a delivery plan to the DDM. DP maintains the agenda and retrieves the first entry of agenda (a goal + subgoal combination) for the next teaching action. Since each individual subgoal belongs to a fixed teaching style (e.g., s_for is an inquiry subgoal, no matter which goal it is associated with), the DP could easily decide which sub-planner it should switch to. The domain programmer specifies the subgoals for each goal, thus the domain programmer in fact decides the selection of sub-planner directly (by considering which piece of knowledge is most suitable to be delivered by which kind of teaching environment). The DP, though providing the switching mechanism, does not define a set of rules to make the decision of sub-planner switching.

The sub-planners select a “delivery style” or a strategy (in the context of the relevant teaching styles) to be used in pursuing a goal, and passes the combinations of goal + subgoal + strategy (+ case, if any) to DDM. There are two sets of selection heuristics associated with each sub-planner, namely, meta heuristics (MHs) and default heuristics (DHs). MHs have the highest priority whereas DHs serves as “bottom lines” that could be superceded by domain-dependent preferences (DDPs) specified from the CK. So the order of the selection heuristics from higher to lower priorities is: MHs - DDPs - DHs.

So far, we have designed three sub-planners; namely: the expository sub-planner, the assessment sub-planner, and the inquiry teaching (sub-)planner (ITP). ITP, the most important delivery sub-planner in TAP-2, will be described in the next section. We have also proposed tentative planning rules (subjected to formalization) for the two basic sub-planners, the expository sub-planner and the assessment sub-planner (these are not the main focus of the TAP project).

The expository sub-planner has four subgoals associated with: s_smrc, s_int, s_pre, s_inb. Four expository styles are identified which are notated as p_des (textual description with or without multimedia aid), p_sim (simulation or demonstration), p_exp (expository through experiment), p_inb (inform the student of his bug). The MHs in the sub-planner are:

- PM1:** Always assign p_inb to s_inb.
- PM2:** Never assign p_sim and p_exp to s_int or s_smrc.

The reason behind PM1 is obvious. PM2 is added because p_sim and p_exp are time-consuming and not suitable for introduction to, summary or recap of a piece or pieces of knowledge.

There is no DH associated with the sub-planner. Domain programmers will be required to specify their own preferences on the expository style for each particular expository subgoal. Specifications that violate the MHs will be ignored by the system.

There is only one type of subgoal that belongs to the assessment sub-planner: s_ase. s_ase could be pursued at any point of the instruction. Five assessment styles have been identified for TAP-2: a_obj (objective questions and yes-no questions), a_fil (fill in the blanks), a_sqa (simple question-and-answer), a_sim (question during the simulation), a_exp (question during the experiment). We have also worked out the only MH for this sub-planner.

- AM1:** Never use a_sim or a_exp for the assessment in the situation of recap.

Again the consideration is that a_sim and a_exp may be too time-consuming for the purpose of “recap assessment”.

At the end of the execution of the first item on the agenda, the next item will be planned and executed, unless student bugs are encountered. In the latter case, the delivery planner will decide which particular action to be carried out to remedy the situation by referring to the following rules,

- DB1:** IF the goal is to teach fact(s),
OR the goal has not been achieved yet AND the goal is not residing on the currently selected goal path,
OR the goal corresponding to the bug has been achieved before AND the system has used up all the suitable teaching strategies to “debug” it,
THEN add s_inb onto the agenda.
- DB2:** IF the goal corresponding to the bug has been achieved before OR the goal is residing on the currently selected goal path,
AND the bug is an incorrect hypothesis,
AND there are still suitable debugging strategies that have not been used before,
THEN add s_deb onto the agenda.
- DB3:** IF the goal corresponding to the bug has been achieved before,
AND the bug is a conceptual error other than incorrect hypothesis,
AND there are still suitable debugging strategies that have not been used before,
THEN add s_rem onto the agenda.

There are three disjunctive conditions for executing DB1. This is the simplest way of handling a bug, s_inb (inform a student). Firstly, it is not worthwhile to create a complex “debugging” subgoal to remove a wrong fact. Next, the unachieved goal will probably be pursued later (so the system can avoid the redundancy of debugging the goal and then teaching it again). If the goal is not selected by the end of the course, the current decision of not dealing it in detail would be more rational. The final condition occurs when the system has tried to re-execute a debugging subgoal by selecting all suitable strategies but the student still does not understand the concept. In such a worst situation, the system could do nothing but execute s_inb.

DB2 and DB3 are simply the rules for retrying the “debugging” goal. s_deb corresponds to a step within the systematic reasoning process - to modify a hypothesis after it is proven incorrect by a testing process. For debugging other conceptual errors, the system will select s_rem.

If more than one bug is identified at the same time, or there are previous bugs which have yet to be removed, the DP has to order the debugging subgoals according to certain priority rules as shown below,

- OB1:** The debugging subgoal corresponding to the most recent bug comes first.
OB2: Errors before omissions.
OB3: Prior steps before later steps (steps in a procedure or causal orders of two concepts).
OB4: Compare the preference ratings of the debugging subgoals.

The ordered set of debugging subgoals will be added in front of the agenda and the planning-execution cycle continues accordingly.

ITP: Inquiry Teaching (Sub-)Planner

The basic responsibility of the ITP is to select a suitable strategy and case(s) for each inquiry subgoal. The ITP in TAP-2 adopts the 59 inquiry strategies in (Collins & Stevens, 1982b), with certain adaptations in order to fulfill some constraints as well as Collins’ other viewpoint (Collins, 1995).

Dialogue Sessions

One significant amendment to the TAP architecture which departs from Collins & Stevens’ theory is to treat the dialogue session of each goal + subgoal combination (no matter whatever the subgoal is an inquiry or non-inquiry one) as the most basic unit for performing instructional planning. This differs from Collins & Stevens’ work whereby the planning and replanning are performed at the lowest level - which occurs each time after the student has “spoken” something and therefore the system responds (e.g., in Figure 3, almost every statement made by the tutor is an application of one or several inquiry strategies).

Such an amendment is made because of two considerations. Firstly, the context switching between the planner and the executor presents a computational overhead. Thus, it is not worthwhile to re-plan the dialogue at the lowest level (that is, high frequency of replanning). Secondly, and more importantly, the modified policy is expected to produce more systematic and modularized dialogue. In this case, the student will have a better idea of the dialogue flow (that is, what are the goals that he has gone through and what is the current goal) and less likely to be put off by the dialogue.

Through the analysis of the excerpts of inquiry dialogue provided by various sources (Collins, 1987; 1988; Collins & Stevens, 1982a; 1982b; Wong, 1994), we induced a phenomenon: the first rule (one of the 59 strategies) that is applied during each dialogue session is always the pre-dominant rule of that session. The choice of “the first rule” decides the general content of that session. We refer to such rules as “Dialogue Session Rules” (DSR). For example, in Figure 3, the system chooses CSS6 (differentiation exemplar of factors) as the DSR of the session. Hence, the entire session is focussed on the discussion of the positive (Yangtze Plain) and negative (Nigeria) exemplar pair.

Once the DSR has been applied to the session, the choice of subsequent rules will depend on the student’s response. Generally, these rules are “small” and could be re-used by more than one dialogue session. ENS2, IS3, ES5, IS7 and ES2 in Figure 3 illustrate such rules. Among them, ES2 has been used twice. This type of rules are known as “Dialogue Micro-Rules” (DMR).

We divide the 59 inquiry rules into two groups: DSR and DMR, based on the nature of each individual rule. Figure 8 shows the categorization of these rules.

	CSS	ENS	IS	ES
DSRs	1-16	-NA-	4-6; 11-14	11-14
DMRs	-NA-	1-12	1-3; 7-10; 15-17	1-10

Figure 8. DSRs and DMRs

On the other hand, we recommend domain programmers to encode the DMRs implicitly in the “dialogue templates” (see later section) while they are programming the UI manager.

The conditions and actions of DMRs are generally quite simple. They are essentially the immediate responses to the student’s latest input. Simple software or IF-THEN-ELSE constructs (with the student’s responses as the conditions) can be used to implement these rules.

Issues in Case Selection

As a case-based teaching method, the selection of cases for the dialogue is a significant issue in inquiry teaching. Collins advised the avoidance of changes of cases. This is a primary concern in case selection; and hence, even with old cases, there should not be too much switching.

Most of the DSRs require the selection of cases (only IS4-6 does not require that). According to (Collins & Stevens, 1982b), the cases required for IS11-14 and ES11-14 are restricted to previously chosen ones. For example,

IS12: Ask for differences in factors between similar cases

IF (1) two or more cases have been identified that have similar values on the dependent variable;

THEN (2) ask the student to identify any factors on which the cases have different values.

On the other hand, there are two kind of rules in the CSS* group. The first kind of rules (CSS5-8) require a previous case and the selection of another case (e.g., CSS6 in Figure 3, where Yangtze Plain [the “positive case”], according to [Collins & Stevens, 1982b], should be a case that has been discussed before; also see the description of the version 0 of CSS6 in Figure 9). Whether to choose an “old” case or a new case as the second case is not explicitly specified. The second type of the rules (the rest of the CSS*) require only one case (e.g., CSS14 in section 2 of Figure 3). Again this can be either an “old” or new case.

CSS6: Differentiation exemplar for factors (Minimal Pair)

Version 0 (original version)

IF (1) a student has not identified one or more factors that are relevant to a particular value on the dependent variable, AND

(2) there is a case identified that is a positive or negative exemplar of those factors,

THEN (3) pick a case that has a different value from the previous case on the given factor, that has the same or similar values on other factors, and that has a different value on the dependent variable.

Version 1

IF (1) a student has not identified one or more factors that are relevant to a particular value on the dependent variable, AND

(2) two cases are available that have different values on the given factor, that have the same or similar values on other factors, and that have different values on the dependent variable,

THEN (3) discuss with the student about the differences of the given factors of the two cases and how they affect the dependent variable.

Version 2

IF (1) the top goal is to teach a necessary factor, AND

(2) two cases are available that have different values on the given factor, that have the same or similar values on other factors, and that have different values on the dependent variable,

THEN (3) pick a case that has a different value from the previous case on the given factor, that has the same or similar values on other factors, and that has a different value on the dependent variable.

Figure 9. Different versions of CSS6

In computerizing CSS*, two important points have to be considered. Firstly, although “old” cases are preferred, we must admit that each of them cannot suit ALL the rules. The action of CSS* (that is, “THEN: pick a case that...”) can serve for the introduction of new cases if the need arises. Hence, a group of MHs for case selection, listed in the order of execution, are proposed as below:

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

- MCS1:** Select the most recently used case(s), if any (cases that have just been used in the latest dialogue session; to ensure the minimal switching of cases).
- MCS2:** Select the most frequently used case(s), if any (in terms of the number of previous sessions that have chosen for each of the cases).
- MCS3:** Select the previously used case(s).
- MCS4:** Select the more common/popular case(s) (among all the unused cases).

This leads us to the second concern: what happens if the DK does not store any case that is suitable for the rule? In this situation, the system is unable to execute the rule since no case can be discussed. In TAP-1, the strategy (a “reduced set” of the inquiry rules in the context of TAP-2) selection and the case selection processes are interleaved - the planner first decides on the strategy and subsequently select suitable cases to the strategy. The planner must “backtrack” to the strategy selection process if no suitable case is found for the selected strategy. This is a significant overhead. The solution in TAP-2 is to promote the actions of “THEN: pick a case...” to be new conditions for relevant rules, and leave the actual dialogue discussions of the cases as the action. Although conceptually the actions of CSS* are case selection, the “case selection actions” of these rules were implemented as conditions in order to improve the performance.

Furthermore, the incorporation of the set of MCS* in ITP makes the restrictions of “selecting only the previous cases” in IS11-14, ES11-14, and CSS5-CSS8 seem insignificant. For instance, whether Yangtze Plain discussed in Figure 3 is a previous case or new case does not affect the dialogue flow. The actual concern is that whether Yangtze Plain is familiar to the student, which is supposed to be taken care of by MCS* in ITP (as MCS* are always activated during every case selection process). Therefore, these restrictions could be omitted from the conditions of the relevant inquiry rules.

The relaxation of these restrictions implies that more strategies can be chosen for every goal + (inquiry) subgoal combination. This particularly benefits the situation whenever TAP-2 keeps trying different strategies in delivering the same goal (when the student still does not understand the piece of knowledge being taught). This tallies with another argument by Collins (1995) that “a good inquiry teacher should keep trying to teach the students (remedy their bugs) whenever is possible”.

Based on all of the considerations presented in this section, CSS6 is modified and becomes version 1 in Figure 9. Whether to select previous case(s) or new case(s) for each inquiry rule is no longer specified at the condition(s) of the rule, but rather decided by MCS* and the availability of the required case(s) of other eligible inquiry rules.

DSR Firing Algorithm

A DSR firing algorithm is proposed in Figure 10 as an effort to organize the rule firing process (note that “:=” means value assignment to a variable while “=” represents comparison).

In the algorithm, the first OR antecedent of the first IF rule in the FOR loop is a criterion inherited from TAP-1. The reason behind this is that it is better to select another inquiry strategy to teach the same goal and subgoal, rather than keep repeating the same goal + subgoal + strategy combination which did not work previously.

The second OR antecedent provides the flexibility for the domain programmer to specify some domain-dependent restrictions in the strategy selection. The restrictions will be specified in CK, in terms of “prohibited subgoal + strategy combinations” (no matter what goal it is) or “prohibited goal + subgoal +strategy combinations”. The reasons behind these restrictions could be because no corresponding dialogue templates are developed, or no teaching material needed for the combinations is available, among others.

The Boolean flag NeedNewCase is another means of rule filtering. This is first initialized with the value “TRUE” and will toggle to “FALSE” only if an eligible strategy; when all its attached cases are re-used ones, has been identified. Hence, all the subsequently identified strategies with new cases will be de-selected. This occurs in two lines labeled (A) in the

algorithm. This filtering process will reduce the execution time of the procedure. Another important step appears at each of the lines labeled (B) and (C) whereby “the most preferred case” is to be decided there. This is where MCS* are activated.

LISTS (global; non-volatile memory)

Unused_Cases := all the cases stored in the domain knowledge base [initially]

Used_Cases := (blank) [initially]

STRUCTURE (local)

Rule_List (eligible rules with their suitable cases)

VARIABLE (local)

Boolean NeedNewCase

PROCEDURE FireDSRs

NeedNewCase := TRUE

Rule_List := (blank)

FOR all the inquiry rules

 IF the goal+subgoal+rule combination has been used in the dialogue before OR
 the combination should not be selected as per domain programmer's specification
 THEN continue with next iteration of the FOR loop (check next rule)
 ENDIF

 check other conditions (i.e., NOT conditions about cases)

 IF other conditions are unsatisfied

 THEN continue with next iteration of the FOR loop

 ENDIF

 IF the rule is CSS* with the requirement of TWO cases

 THEN select the first case from the Used_Cases ---- (B)

 IF no case is found

 THEN continue with next iteration of the FOR loop

 ELSE select the second case from the Used_Cases ---- (B)

 IF a case is found

 THEN NeedNewCase := FALSE

 ELSE IF NeedNewCase := TRUE ---- (A)

 THEN select the second case from the Unused_Cases ---- (B)

 ENDIF

 IF a case is found in either Used_Cases or Unused_Cases

 THEN add the rule and the cases into Rule_List

 ENDIF

 ENDIF

 ELSE IF the rule is CSS* with the requirement of ONE case

 THEN select the case from the Used_Cases ---- (B)

 IF a case is found

 THEN NeedNewCase := FALSE

 ELSE IF NeedNewCase := TRUE ---- (A)

 THEN select the case from the Unused_Cases ---- (B)

 ENDIF

 IF a case is found in either Used_Cases or Unused_Cases

 THEN add the rule and the case into Rule_List

 ENDIF

 ELSE (IS* or ES*)

 select cases from the Used_Cases ---- (B)

 IF cases are found

 THEN NeedNewCase := FALSE

 add the rule and the case into Rule_List

 ENDIF

 ENDIF

 IF Rule_List has more than one rule

 THEN pick a rule with the most preferred case(s) being attached to ---- (C)

 ENDIF

ENDFOR

END PROCEDURE

Figure 10. DSR Firing Algorithm

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

The DSR firing algorithm (for strategy selection process) can be formalized in terms of MHs and DHs as described in the previous sections. The only MH of the process is,

IM1: Never repeat any combination of goal + subgoal + strategy in the instruction.

After all, domain-programmers are allowed to specify a more restricted set of suitable strategies for every goal + subgoal combination in a particular domain.

Three more DHs are specified as below,

ID0: Form the eligible list by firing all the inquiry rules, minus the strategies filtered by IM1 and the programmer-specified preferences.

ID1: IF the eventual eligible strategy list is blank (that is, all the possible combination of goal + subgoal + strategy has been used before),
THEN halt the inquiry planning process and report “planning failure” to DP.

ID2: Select a strategy from the rule list with the most preferred cases being attached to.

The execution of ID1 will subsequently make the DP to activate the rule DB1, that is, to inform the student of the correct answer (s_inb).

Refinements of the rule conditions

Further analysis of the application of the rules to the inquiry dialogue reveals that the conditions of many of the 59 strategies are loosely defined in (Collins & Stevens, 1982b). In actual dialogue, some rules are applicable to certain situations but not others. Thus, we have revised all the DSRs and added in some new conditions. This effort is very similar to what Ikeda & Mizoguchi (1994) have done to the 20 strategies in their FITS (Framework for ITS). They investigated the effects, strength and applicable situations that vary from strategy to strategy (that are incorporated to FITS) and subsequently worked out the major parts of the if-then style E-rules (elaboration rules; that is, strategy selection rules).

Take for example IS12 (ask for differences in factors between similar cases) whose description is shown below:

IF (1) two or more cases have been identified that have similar values on the dependent variable;
THEN (2) ask the student to identify any factor on which the cases have different values.

The only condition as specified above is simple and only involves the required cases. However, from the following example (adapted from [Collins & Stevens, 1982b; pg.112]) one may conclude that such strategy is only applicable to teaching unnecessary factors:

“Supposing that both Japan and Java have been identified as producing rice, the teacher could ask the student for any differences in factors between the two cases. In fact Japan is much more mountainous. This indicates that flat land is not a necessary factor.”

Also, such strategy is more suitable for pursuing the subgoals: s_tes and s_deb, but is less suitable for pursuing s_for. Therefore, a modified version of IS12 is proposed as follows:

IF (1) the top goal is to teach an unnecessary factor
AND (2) the subgoal is either s_tes or s_deb
AND (3) two cases that have similar values on the dependent variable are found,
THEN (4) ask the student to identify any factors on which the cases have different values.

The same modification is made on CSS6 that becomes version 2 in Figure 9.

The expansion in the conditions is not only necessary, but also provides further filtering criteria for rule firing. Since the bottleneck of the condition checking is at the case selection process (particularly if the search space is big, that is, large number of cases is stored in DK), the additional conditions (which are simply value checking of a few individual global variables

like top goal and subgoal) allow possible early rejection of more strategies before the case selection is executed, hence the improvement in performance.

Summary of Modifications Made in ITP

In this section, we have proposed methods to make the ITP in TAP-2 fulfill the following requirements:

1. take “drilling a case to its death” as the main concern of not only the case selection, but also the inquiry strategy selection;
2. inherit TAP-1’s policy of making each dialogue session as a “planning unit”, while at the same time adopting the 59 inquiry strategies which, in contrast, takes each statement of the tutor as a planning unit;
3. reduce the possibly considerable computational time in rule-firing (especially the bottleneck in the case selection process) without the expense of simplifying the specifications of the inquiry strategies.

Summary of Program Flow of TAP-2

The planning-execution cycle of TAP-2 can be summarized as follows:

[GCP]

1. Selects a goal from the list of ultimate goals for current focus.
2. Identify the shortest unblocked learning path leading to the ultimate goal.
3. Selects a goal from the eligible goals (i.e., the goals where all their prerequisites have been achieved) in the selected path.

For example: g_1 is selected

[DP]

4. Breaks the goal into several subgoals, and adds them onto the agenda.
For example: According to the specification in CK, the subgoals attached to g_1 include s_for, s_tes, s_ase, s_smrc (see section 3.1 for the description of these notations). Therefore, the following items are posted onto the agenda:

{ g_1+s_for, g_1+s_tes, g_1+s_ase, g_1+s_smrc }

5. Switches the planning task to the suitable sub-planner, given the first item of the agenda (goal+subgoal).

For example: According to the above agenda, the first item is g_1+s_for.

[ITP] (Assuming that ITP is selected; In case other teaching style is selected, this portion is to be replaced by the planning process of the respective sub-planner).

6. Selects a suitable inquiry strategy.
7. Selects suitable case(s) for the current plan.

[Executor/UI Manager]

8. Executes the plan and diagnoses the student’s response.

[Student Modeler]

9. Updates the SM.

[DP]

10. Creates and orders subgoals to correct bugs if they occur.

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

11. If a debugging subgoal fails and it causes the current path in the global curriculum plan to be blocked, then loop back to (2); else loop back to (5).
12. Loops back to (1) until ultimate goals are all achieved or no more path leading to unachieved goals could be chosen.

At the end of the execution, the inquiry teaching session is considered perfectly or partially successful, depending on how the pre-specified ultimate goals are all or partially achieved respectively. On the other hand, if none of the ultimate goals is accomplished, the session is considered unsuccessful.

A Domain Case Study

We have made use of the TAP-2 architecture to implement PADI-2, a geography tutor. PADI-2 discusses “factors that affect rice growing” and “factors that affect the capability of a place to apply agricultural technology (e.g., irrigation, terracing, mechanization, etc.). For the first portion of the domain, we identified four necessary factors that affect rice growing: water supply (either heavy rainfall or irrigation), flat area (either naturally flat terrain or terraced land), fertile soil, warm temperature, and labour intensity (either large number of farmers or mechanization). For the second portion of it, we specified two necessary factors that affect the capability to apply technology: rich farmers, and good literacy of farmers/no social prejudice against technology.

After the domain analysis is completed (and the information are encoded in DK in terms of semantic network of causal relationships), we specified 13 top-level goals to be pursued in PADI-2. The descriptions of and the subgoals attached to the goals are:

- g_INTRO**: introduction to the entire course (s_int)
- g_FINDRG**: find out what the students knows about factors affecting rice growing (s_fin)
- g_WATER**: water supply as a necessary factor for rice growing (s_for; s_tes)
- g_FLAT**: flat area as a necessary factor for rice growing (s_for; s_tes)
- g_SOIL**: fertile soil as a necessary factor for rice growing (s_for; s_tes)
- g_WARM**: warm temperature as a necessary factor for rice growing (s_for; s_tes)
- g_LABOR**: high labour intensity as a necessary factor for rice growing (s_for; s_tes)
- g_SUMRG**: summary of factors that affect rice growing (s_pre)
- g_FINDTEC**: find out what the student knows about factors affecting the application of technology (s_fin)
- g_FINANC**: rich farmers as a necessary factor for the application of technology (s_for; s_tes)
- g_SOC**: good literacy and no social prejudice as a necessary factor for the application of technology (s_for; s_tes)
- g_SUMTEC**: summary of factors that affect the application of technology (s_pre)
- g_SUMALL**: summary of the entire course (s_pre)

These together with the prerequisite relationships between the goals are specified in CK.

Figure 11 depicts the resultant global curriculum network. The ultimate goals of the domain are specified here (presented as bold ellipses in Figure 10): **g_SUMRG**, **g_SUMTEC**, and **g_SUMALL**. This is to ensure that the set of goals related to each of the two dependent variables (rice growing and the application of technology) will be attempted. The curriculum network of PADI-2 is relatively simple since there is only one learning path leading to each of the ultimate goals.

Note that the prerequisites of **g_FINDTEC** consist of **g_WATER**, **g_FLAT** and **g_LABOR** instead of **g_SUMRG**, although it is desired to discuss all of the factors that affect rice growing (and summarized by **g_SUMRG**) before the tutor moves on to the factors that affect the application of technology. This is because only **g_WATER** (involving irrigation), **g_FLAT**

(involving terracing) and g_LABOR (involving mechanization) are relevant to the application of technology.

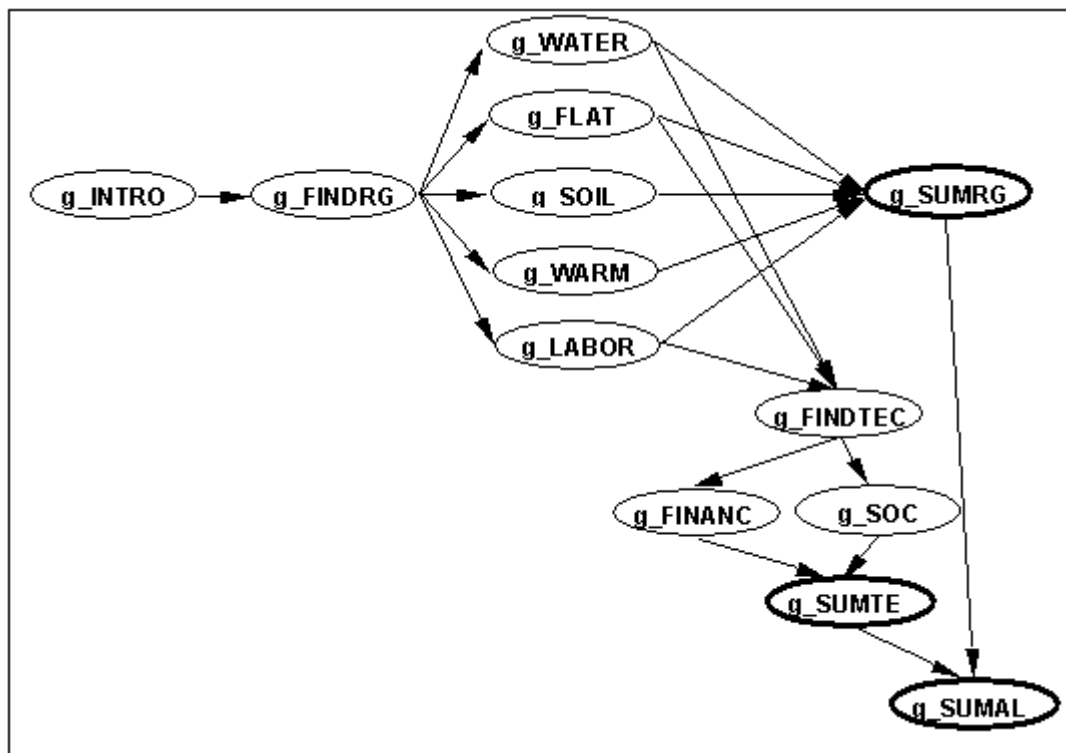


Figure 11. Global Curriculum Network of PADI-2

Two groups of “debugging top goals” are also specified in CK. The first group is concerned with specific irrelevant factors that include g_IRORIEN (oriental nature) and g_IRTRPCL (tropical climate). These goals will be made eligible if and only if the student has committed the relevant bug - that is when she thought that oriental nature or tropical climate is necessary for rice growing. More “debugging top goals” corresponding with common bugs can be added once they are identified.

The second group of “debugging top goals” is relevant to the omission of unnecessary factors. For instance, a student thought that heavy rainfall is necessary for rice growing, implying that she does not know irrigation is an alternative to heavy rainfall. There are six goals of this kind in PADI-2: g_OMTRAIN (omit rainfall); g_OMTIRR (omit irrigation); g_OMTNFLT (omit naturally flat terrain); g_OMTTRCG (omit terracing); g_OMTMPWR (omit manpower); g_OMTMECH (omit mechanization).

Specification of Cases

Cases are important means in the inquiry dialogue. To date, 102 cases (various places in the world) have been identified and encoded in DK of PADI-2. Each case is assigned a preference rating (in accordance with Singapore student’s familiarity with the place) which is again represented by an integer between 0-511 (higher number indicates more preferred cases). Two important cases with their attached information are shown here as examples:

- Bangladesh - preference rating: 500
- has heavy rainfall, irrigation facilities, water supply, naturally flat terrain, flat area, fertile soil, warm temperature, strong manpower, and high labour intensity;
- does not mechanize rice growing, has no terraced land;

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

- an grow rice
 - has oriental nature and tropical climate (irrelevant factor for rice growing)
 - has no rich farmers, has poor literacy and prejudice; cannot apply agricultural technology

- Switzerland - preference rating: 450
 - has irrigation facilities, water supply, fertile soil, strong manpower, and high labour intensity; does not mechanize rice growing, has no heavy rainfall, has no terraced land, has cold temperature; cannot grow rice
 - has no oriental nature and tropical climate
 - has rich farmers, and good literacy and no prejudice; can apply agricultural technology

Note that in real life it is inappropriate to represent the existence of the factors in each of the places by “yes” or “no”. The logical representation is just a matter of convenience for the system to handle the dialogue. For instance, to say Bangladesh “has no rich farmers” actually means that the amount of rich farmers is probably negligible in Bangladesh.

Dialogue Templates

In PADI-2, we have attempted to implement every possible combination of subgoal + strategy as a “dialogue template”. A dialogue template is a C procedural function using if-then and case-switch constructs to generate and handle the inquiry dialogue. Depending on the students’ response at each relevant point, the corresponding switch will branch the dialogue to some “sub-template”. In a way, the dialogue templates are similar to the general AI knowledge representation of “script”.

Each dialogue template accepts the current goal and the selected case(s) (decided by the planner) as input parameters. The description of the goal and the name of the case (a place) in the real world will be substituted into various points of the printf() statements in the dialogue template, so they could be presented to the student. For example, Session 1 in Figure 3 is handled by the s_for + CSS6 template (CSS6 is the pre-dominant strategy in this session as it generates the first question being asked by the tutor) with g_WATER (the goal), and Yangtze Plain & Nigeria (cases) as the parameters.

The construct of dialogue template is one possible way of implementing a discourse manager in a TAP-based tutor. It is not a compulsory method for future domain programmers to follow.

Discussions

In short, the process of implementing an inquiry tutor by using the TAP-2 architecture can be summarized in the following steps: (1) domain analysis; (2) design and encoding of DK and CK; (3) development of executor (that usually involves a discourse manager or dialogue templates) and UI manager; (4) integration of the domain-specific module and the TAP-kernel.

It is not our intention to develop a system that concentrates on assessing the student. In promoting the student’s ability in systematic thinking, the system monitors the student in constant engagement with the systematic process (“practice makes perfect”). The process is repeated by pursuing many inquiry top-goals in the curriculum. The delivery of the debugging subgoals is part of the knowledge discovery/construction process, rather than just a remedial action. Therefore, in dealing with the student’s bugs, TAP does not carry the attitude of “Ah hah! You are wrong [again]!” but rather “Okay! We have made a guess (formed a hypothesis). Now let us test it...” That is, correct hypotheses and wrong hypotheses are treated in the same

way in terms of the words used during the discussions, although the sets of inquiry strategies that are applied to “test a hypothesis” and “debugging an incorrect hypothesis” respectively are not identical. Of course, helping the student to understand the conceptual/causal knowledge in greater depth is a side-benefit of such a teaching style.

EVALUATIONS

To date, PADI-2 has been demonstrated to an educational specialist working in the gifted-school programme from the Ministry of Education, Singapore; who is in charge of the design of courses to train “inquiry teachers” at the National Institute of Education, and 7 local lecturers and teachers who have been practicing inquiry teaching between 5 and 15 years. They all agreed that the dialogues generated by PADI-2 are coherent inquiry dialogues and they believe that the approach is effective to the students. A full report of the interview is available in (Wong, 1998).

Further, we have also developed a fraction tutor, called FT-TAP, based on the TAP architecture (Lim, 1997). FT-TAP underwent a full-scale evaluation that involved about 180 students with favourable comments by the students on the quality of instructions provided by the tutor (Sing, 1998). Work is currently being devoted to the development of a full-fledge WEB tutor using the TAP kernel. Two experimental WEB-sites are available to support PADI-2 and FT-TAP. They are:

PADI-2: <http://uranus.sas.ntu.ac.sg:8000>

FT-TAP: <http://uranus.sas.ntu.ac.sg:8001>

CONCLUSION

TAP is intended to demonstrate an inquiry tutor that is developed within the context of ITS. The major contribution is the investigations of the feasibility of and issues in computerizing inquiry teaching. A TAP-based ITS should be able to repeatedly engage students in the systematic reasoning process by guiding them to re-construct numerous rules and concepts. With the aid of a curriculum planning technique, these rules and concepts can be sequenced in an organized manner.

Computationally, the integrated planner in TAP-2 is a combination of two general AI planning techniques: state-space and rule-based planning. The entire DP is designed as a rule-based planner - the planning decisions of each sub-module is made by referring to the particular set of rules attached to it. GCP is essentially a state-space (STRIPS-like) planner, although its planning policy is controlled by some meta rules.

In terms of instructional planning, the planner in TAP-2 follows the opportunistic plan-based approach (or dynamic instructional planning), in which plan generation and revision occur during the instructional session in response to the changing tutorial situation (Murray, 1989). The combination of curriculum planning and delivery planning, as what TAP-2 has achieved, is one of the hottest issues in current ITS community.

An educational software incorporating TAP is intended to allow one to conduct a one-to-one inquiry session with individual students, which is almost impossible to be achieved by human inquiry teachers due to the low information-transfer rate of such teaching style. Another threat of the human-delivered inquiry method that Collins & Stevens (1982b) have pointed out was the required level of sophistication of the teacher - the teachers must be able to think quickly in handling all the opportunistic situations. An effective planning mechanism that is implemented as a software will be able to make use of the fast computational power of the ever improving hardware.

Future work in the development of TAP-2 includes:

Tap-2: A Framework for an Inquiry Dialogue Based Tutoring System

1. the investigation of other scientific inquiry methods (Collins & Stevens' theory represents one of them) as which components can be supported or automated by technology, or adopted by TAP to enhance the system;
2. the development of a full-fledged "inquiry-centered" tutor with the incorporation of other BSTPs which are based on formal frameworks;
3. the use of the TAP architecture to construct a test bed for the human teachers to experiment with the variations in inquiry dialogue, such as the effects of different planning rules, choices of goals, subgoals and strategies;
4. the investigations of incorporating inquiry teaching into various learning environments (e.g., Learning Companion Systems [Chan et al., 1992], cognitive apprenticeship [Collins et al., 1989], mindtools, simulations, etc.), by making use of TAP-2 architecture.

Acknowledgements:

We wish to thank Allan Collins for discussions on inquiry teaching.

References:

- Brecht (Wasson), B.J. (1990), Determining the focus of instructions: Content planning for intelligent tutoring systems, *Ph.D. Dissertation*, Department of Computational Science, University of Saskatchewan, Saskatoon, Canada, June 1990.
- Chan, T.W., I.L. Chung, R.G. Ho, W.J. Hou, & G.L. Lin (1992), Distributed learning companion system: West revisited, In: C. Frasson, G. Gauthier, G.I. McCalla (Eds.), *Lecture Notes in Computer Science: Intelligent Tutoring Systems*, Berlin, Germany: Springer-Verlag.
- Collins, A. (1987), A sample dialogue based on a theory of inquiry teaching, In: C.M. Reigeluth (Ed.), *Instructional Theories in Action: Lessons Illustrating Selected Theories and Models*, Hillsdale, N.J.: Erlbaums, 181-199.
- Collins, A. (1988), Different goals of inquiry teaching, in *Questioning Exchange*, 2(1), 39-45.
- Collins, A. (1995), *personal conversation*.
- Collins, A., J.S. Brown, and S. Newman (1989), *Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics*, In: L.B. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 453-494.
- Collins, A., and A.L. Stevens (1982a), A cognitive theory for inquiry teaching, In: P. Goodyear (Ed.), *Teaching Knowledge and Intelligent Tutoring*, Norwood, N.J.: Ablex, 1991, 203-230.
- Collins, A., and A.L. Stevens (1982b), Goals and strategies of inquiry teachers, In: R. Glaser (Ed.), *Advances in Instructional Psychology II*, Hillsdale, N.J.: Erlbaum, 1982.
- Collins, A., E.H. Warnock, and J.J. Passafiume (1975), Analysis and synthesis of tutorial dialogues, In: G.H. Bower (Ed.), *The Psychology of Learning and Motivation*, 9, N.Y.: Academic Press.
- de Castro, M.I., A. Sanchez, and M.F. Verdejo Maillo (1988), Building a programming tutor by dynamic planning: Case studies and a proposal, *Proceedings of ITS-88*, Montreal Canada, 230-237.
- DeBoer, G.E. (1991), *A History of Ideas in Science Education*, N.Y.: Teachers College Press, 206.
- Derry, S.J., L.W. Hawkes, and U. Ziegler (1988), A plan-based opportunistic architecture for intelligent tutoring, *Proceedings of Intelligent Tutoring Systems (ITS-88)*, Montreal, Canada, 116-123.

- Elsom-Cook, M. (1988), The application of machine learning to intelligent tutoring systems, In: J. Self (Ed.), *Artificial Intelligence and Human Learning, Intelligent Computer-Assisted Instruction*, N.Y.: Chapman and Hall, 179-196.
- Hietala, P. (1989), Applications of AI in education: An overview, *Proceedings of Scandinavian Conference on AI-89*, 299-306.
- Ikeda, M., & R. Mizoguchi (1994), FITS: A framework for ITS - a computational model of tutoring, *Journal of AI-ED*, 5(3), 319-348.
- Lim, C.J. (1997), FT-TAP: An inquiry dialogue based fraction tutor for constructive learning, *Internal Report*, School of Applied Science, Nanyang Technological University, Singapore, 1997.
- MacMillan, S.A., D. Emme, and M. Berkowitz (1986), Instructional planners: lessons learned, In: J. Psozka, D. Massey and S. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned*, N.J.: Lawrence Erlbaum Associates.
- Murray, W.R. (1989), Control for intelligent tutoring systems: A blackboard-based dynamic instructional planner, *Proceedings of AI-ED'89*, Amsterdam, Netherlands, 150-168.
- Ohlsson, S. (1986), Some principles of intelligent tutoring, *Instructional Science*, 14, 293-326, 1986. Also In: R. Lawler and M. Yazdani (Eds.), *AI and Education: Learning Environments and Intelligent Tutoring Systems*, N.J.: Ablex Publishing, 1987.
- Peachey, D.R. (1983), An architecture for plan-based computer assisted learning, *M.Sc. Dissertation*, Department of Computer Science, University of Saskatchewan, Saskatoon, Canada.
- Peachey, D.R., and G.I. McCalla (1986), Using planning techniques in intelligent tutoring systems, *Int. Journal of M-M Studies*, 24, Nor. 1-6.
- Peterson, P.L., W.R. Marx, and C.M. Clark (1978), Teacher planning, teacher behavior and student achievement, *American Educational Research Journal*, 15(3), 416-437.
- Rakow, S.J. (1986), Teaching science as inquiry, *Fastback 246*, Bloomington, I.N.: Phi Delta.
- Sing, S.H. (1998), A controlled evaluation of a TAP-based inquiry fraction tutor, *Internal Report*, School of Applied Science, Nanyang Technological University, Singapore, 1998.
- Steven, A.L., and A. Collins (1977), The goal structure of a Socratic tutor, *Proceedings of Association of Computing Machinery Annual Conference*, Seattle, W.A.
- Wong, L.H. (1994), An agenda planner for inquiry teaching, *Internal Report*, School of Applied Science, Nanyang Technological University, April 1994.
- Wong, L.H. (1998), A structured system's view of an intelligent tutoring system based on inquiry teaching approach, *Ph.D. Dissertation*, School of Applied Science, Nanyang Technological University, August 1998.
- Wong, L.H., H.C. Quek and C.K. Looi (1998), TAP: A software architecture for an inquiry dialogue based tutoring system, *IEEE Transaction on System, Man and Cybernetics*, May 1998.
- Zahorik, J.A. (1975), Teacher's planning models, *Educational Leadership*, 33(2).