PhD Dissertation Proposal (draft)

Is Proactive Scaffolding Worth Forcing on Students Learning Theorem Proving?

Noboru Matsuda
Intelligent Systems Program
University of Pittsburgh

August 5, 2003

1. Introduction

The purpose of the current study is to investigate an effectiveness of an intelligent learning environment designed to support students to learn geometry theorem proving. A novelty of our intelligent tutor lies in the way it assists students learning how to write a proof. Unlike many existing tutors that expect students to solve problems by themselves and provide help only at impasses, our tutor offers students with intensive articulation of the problem-solving steps in detail. Only after the tutor believes that the student reaches a certain level of competence, then does the tutor gradually fade the amount of articulation and further switch the form of scaffolding from articulation to inquiry. We call this kind of assistance *proactive scaffolding.*

This simple form of teaching strategy involves several interesting aspects of cognitive theories on teaching and learning.

We have designed an intelligent tutoring system that can provide proactive scaffolding. The ultimate goal of this dissertation study is to test specific research questions regarding the effectiveness of such tutoring strategy by analyzing students' learning outcome. To answer those questions, we will run experiment with an intelligent tutoring system for geometry theorem proving that is called *AdvGeo tutor*.

This proposal first discusses current state of the art of effective teaching strategies and students' difficulties in teaching and learning geometry theorem proving. We then describe existing intelligent tutoring systems for learning theorem proving and discuss issues in such systems. Section 2 then discusses cognitive theories on learning and teaching theorem proving that provides insight into the research questions asked in this study, which is asked in Section 3. Section 4 describes the AdvGeo tutor designed for the current study. The description of the ongoing intelligent tutoring system consists of cognitive task analysis for geometry theorem proving, proactive scaffolding strategy, and the individual component in the learning environment to help students learn how to write a proof. Section 5 is about the experiment to address the research questions, especially on the hypothesis discussed in section 5.1. The proposal conjectures possible contributions of the current study to the research community on intelligent tutoring systems in Section 6. Finally, a chronological estimation to accomplish current dissertation study is provided in Section 7.

2. Overview of Teaching and Learning Problem Solving and Theorem Proving

2.1. Students' Difficulties in Learning Theorem Proving

Geometry theorem proving is one of the most challenging subjects for students. They fail to write proofs even when they are competent for concepts on geometric propositions and postulates. Koedinger (1991) reported that the students had only 35.5% accuracy on proof writing at pre-test even though they had 67% accuracy on judgment of geometric statements. In a large scale evaluation with 1520 students from 74 classes in 11 schools, Senk reported that 30% achieved about 75% masterly level, 40% had some proof skills, and 30% gained no competence on proof writing at all. No more than half the students were capable of a non-trivial proof at the end of school year (Senk, 1983). In Thailand, less than 15% of students achieved "Good" in proof writing even though they are assessed as the van Hiele level of 4 (which means …) (Chaiyasang, 1989).

2.2. Intelligent Tutoring System for Teaching Theorem Proving

One common strategy to make students capable of writing a proof is to let them solve many proof problems while providing feedback and help when needed. Several Intelligent Tutoring Systems (ITSs for short) have been developed for this purpose.

2.2.1. Intelligent Tutoring System for Teaching Problem Solving Steps

Before intelligent tutoring systems had invented, traditional computer assisted instruction (CAI) systems simply provided a series of *problems* for students to solve. When a student fails to solve a problem, then the CAI system provides feedback and/or other problems until the student achieves mastery. In Figure 1 a), a sequence of problems is represented as P_1, P_2, …, P_k.

Intelligent Tutoring Systems, especially model tracing tutors, tend to articulate problem solving *steps* and diagnose student's competence to perform these steps. In Figure 1 b), S_{ij} shows j-th step applied to solve problem P_i. This type of tutors can provide feedback and helps for each step during the student's trial of problem solving. Like traditional CAIs, these tutors continue to provide problems until the student reach mastery.

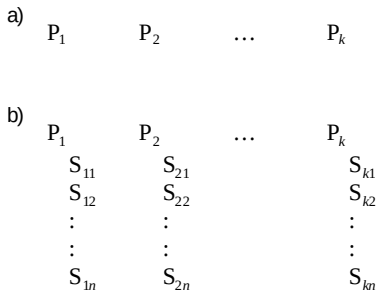

Figure 1: Cognitive Skills taught in traditional computer mediated learning systems

Model tracing tutors assume that the students have somehow acquired required knowledge to perform steps involved in solving problems. The tutor's goal is then to improve the accuracy of knowledge and strengthen its activation.

2.2.2. Feedback and Hinting

Both traditional CAI systems and ITSs utilize the same strategy on feedback and hinting. They basically provide assistance when the students make a mistake or show erroneous performance. These kinds of assistance are used to diagnose and remedy student's misconceptions and errors.

2.2.3. Issues in the traditional learning environment

Above tutors, both traditional CAIs and ITSs, assume that the students have acquired domain principles somehow prior to the tutoring sessions.

2.3. Cognitive Theories for Teaching and Learning Theorem Proving

Initial to intermediate learning: learning domain principles in a declarative form, turning them into procedural form (VanLehn, 1996). Operationalization (Anderson, 1987).

Many theories available to explain

Strategy seldom taught.

2.3.1. Learning from Examples and Problems

Self-explanation

Problem solving,

Cognitive load: guided problem solving

2.3.2. Goal Reification

2.3.3. Operationalization

Anderson

Tubridy (1992)

3. Research Questions

We hypothesize that exposing intensive explanation on problem solving steps and repetitious practice on them facilitates initial learning, especially learning domain principles required solving problems.

To test these hypotheses, we have designed an intelligent tutoring system that provides proactive scaffolding and guided problem solving environment.

The tutor shows students how to solve problems by articulating even the inferences steps that are usually processed within the problem solver's mind hence not explicitly taught. The tutor has a model of problem solving that represents not only applications of domain principles but also the "hidden" inference steps invoked to apply domain principles. Our first hypothesis is that this proactive articulation brings better learning gain in the latter problem solving phase than letting students solve problem without going through proactive scaffolding. Our second hypothesis is that having finer cognitive steps (i.e., underlying inference steps to apply domain principle) is better than just articulating a sequence of application of domain principles.

Once the students become familiar with the domain principles, they need to strengthen their cognitive skills to apply those principles in an actual problem-solving situation. The repetitious practice helps this process, but at the beginning, the students reach impasse that impede learning. Our third hypothesis is that guided problem-solving environment, which continuously provides students with the steps to follow, never allows a student to reach an impasse hence results in better learning.

In sum, we answer following research questions in the current dissertation study:

- Does learning in the fully proactive scaffolding environment bring "better" learning gain than learning with the traditional intelligent tutoring system in terms of the amount of transfer and learning time?

- Does providing proactive scaffolding on the "hidden" inference results in better learning?
- Does providing guided problem-solving prior to an ordinal problem solving session results in better learning?

Since the proposed learning environment offers such dense information to students, not many students would be willing to study with our tutor. Additional research question might be worth asking:

- Is the tutor with such thorough scaffolding acceptable for students?

The acceptability is measures by questionnaires, drop rate between pre- and post-test, or the do-you-wanna-use-this-gain test.

4. AdvGeo Learning Environment

This section describes the Advanced Geometry Tutor (AdvGeo for short) that is an intelligent tutoring system designed to assist students to learn geometry theorem proving.

4.1. Cognitive Task Analysis for Theorem Proving

It is important to *know* exactly what should be taught to students. This section provides a cognitive task analysis for geometry theorem proving.

Theorem proving involves to basic skills: forward inference to make an assertion based upon known facts and backward inference to make a chain of deduction from goal to prove towards known facts. Students must learn these two skills hence the tutor needs a cognitive model that represent these skills.

We have analyzed forward and backward inferences in finer cognitive steps than just an application of a rule. Following two sections describe those forward and backward inferences. The model shows not only the inference steps but also a possible articulation and inquiry used for scaffolding on each step, which is described in Section 4.2.

4.1.1. Forward Inference

4.1.2. Backward Inference

Apply Backward Inference

- *Articulation: "Now, you need to make a backward reasoning."*

- *Inquiry: "Do you remember how to make a backward reasoning?"*
- *Associated configuration: None*

Select a goal

- *Articulation: "Select a goal to prove next."*
- *Inquiry: "Can you pick up a goal to prove next?"*
- *Associated configuration: None*

Recognize goals that have not yet proved

- *$goalQueue ¬ goals that have not been proved yet*
- *Articulation: "You need to prove $goalQueue"*
- *Inquiry: "Do you remember which goals must be proved?"*
- *Associated configuration: None*

Select a goal to prove first

- *$goal ¬ first goal in the goal queue*
- *Articulation: "Select a goal to prove first from $goal-queue"*
- *Inquire: "Which goal would you try first?"*
- *Procedure: "Let's try to prove $goal first."*
- *Associated configuration: None*

Goal test

- *Articulation: "Need to see if $goal is known to be true or not"*
- *Inquiry: "Is $goal known to be true?"*
- *Procedure: If $goal is true then "$goal is known to be true, so we have done on $goal" else "$goal is not known to be true yet, so we need to prove it."*
- *Associated configuration: None*

Apply rule

- *Articulation: "Apply known theorem backwards to prove $goal"*
- *Inquiry: "How do you prove $goal?"*
- *Associated configuration: None*

Select a rule

- *Articulation: "Find a rule that has a same consequence as the goal."*
- *Inquiry: "Can you find a rule that has the same consequence as the goal?"*
- *Associated configuration: None*

Transform postulate into operational rules

- *$rule ¬ the target rule*
- *$subgoals ¬ subgoals of $goal*
- *Articulation: "Transform $rule into an applicable form."*
- *Inquiry: "Can you transform $rule into an applicable form?"*
- *Associated configuration: None*

Instantiate consequence

- *Articulation: "See if the consequence of $rule matches with $goal"*
- *Inquiry: "Can you tell if the consequence of $rule? matches with $goal?"*

Identify the consequence

- *$consequence ¬ consequence of $rule*
- *Articulation: "Identify the consequence of $rule "*
- *Inquiry: "What is the consequence of $rule?"*
- *Procedure: "Rule $rule has $consequence as its $consequence"*
- *Associated configuration: None*

Unify the consequence

- *Articulation: "Match $consequence with $goal"*
- *Inquiry: "Can you overlap $consequence to $goal?"*
- *Procedure: "$consequence matches with $goal"*
- *Associated configuration: None*

Execute the rule

- *Articulation: "Now execute $rule"*
- *Inquiry: "Can you execute $rule?"*

Instantiate premises

- *Articulation: "Need to know what we need to conclude $goal."*
- *Inquiry: "Can you tell me the premises of $rule?"*

Identify premises

- *$premises ¬ premises of $rule*
- *Articulation: "Identify premises of $rule "*
- *Inquiry: "Can you identify the premises of $rule?"*
- *Procedure "The rule $rule has $premises as its premises."*

Unify the premises

- *Articulation: "Unify $premises with the problem and see what we need to prove $goal."*
- *Inquiry: "Can you express $premises in terms of the problem statement?"*
- *Procedure "$premises correspond to $subgoals in this problem."*

Check goal iteration

- *Articulation: "Need to see if the subgoals $subgoals has been aimed before "*
- *Inquiry: "Has $subgoal not been mentioned yet?"*
- *Procedure: "The subgoals $subgoals has not been aimed yet."*

Assert the premises of the rule as subgoal

- *Articulation: "Push the subgoals $subgoals onto the goal-queue"*
- *Inquiry: ""*
- *Procedure: "The premises of the rule $rule must be added to the goal-queue. So, you need to prove $goal-queue"*

4.2. Proactive Scaffolding

In this proposal, a problem-solving *step* corresponds to a postulate application for a single inference, either forwards or backwards. A postulate application requires several cognitive skills including, for example, identifying premises of the postulate, unifying them with the problem, and asserting new propositions, and so on. These cognitive skills to perform a problem-solving step are called *inference*.

AdvGeo tutor starts from showing all inferences underlying each step to solve a given problem. An inference can be exposes to students in different ways depending on the student's competence level. For example, for a student who is not familiar with the target inference, the tutor can provide detailed articulation that directly explains how to perform the inference. For a student who is at an intermediate level, the tutor would rather ask a question about the inference so that the student can remember the previous articulation and perform the inference by himself.

Unlike other tutors, the AdvGeo tutor provides scaffolding proactively even when it is not an impasse situation. And such *proactive scaffolding* fades as the student's competence level increases. More precisely, the tutor follows scaffolding strategy shown in Figure 2.

```
Procedure scaffolding( problem )
     solution-tree ← a linear solution tree for problem
     for each step in solution-tree do
          if step is a backward step
               scaffolding-step( :backward-inference )
          else /* step is a forward step */
               scaffolding-step( :forward-inference )

Procedure scaffolding-step( inference-step )
     competence-level ← the student's competence level for inference-step
     if competence-level is RUDIMENT do
          articulate inference-step
          sub-inference-steps ← inference steps required to achieve inference-step
          for each subinference in sub-inference-steps do
               bc-scaffolding-step( subinference )
     if competence-level is INTERMEDIATE do
          inquiry inference-step
     /* if competence-level is STABLE then do nothing */
```

Figure 2: Scaffolding Strategy

Given a problem, the tutor first generates a *solution tree*, which represents process of search for a proof, exclusive of the inferences that are not involved in the proof. For backward chaining, this means that backing up at an impasse on a garden path is excluded from the solution tree. For forward chaining, deduction of a proposition that does not appear in the proof is excluded. Intuitively, the tutor can generate a solution tree by traversing a *proof tree* (an AND-tree) depth-first. Hence, a solution tree is a linear directed graph. Since the nodes in a solution tree roughly correspond to the states in a search space, a node in a solution holds (a) a current goal to prove, (b) a goal queue, (c) a goal stack, and (d) subgoals to achieve the current goal. Figure 3 shows an example of translation of a proof tree into a solution tree.

Figure 3: Generation of a solution tree from a proof tree

The AdvGeo tutor has a simple student model that represents the student's level of competence for each inference. The competence is represented in one of three discrete value: *rudiment*, *intermediate*, and *stable*. The competence level of an inference increases when the student shows a correct performance on it, and decreases when he shows an incorrect performance. As shown in Figure 2, when the competence-level of an inference is "rudiment," then the tutor provides full scaffolding including an articulation of the inference and further scaffolding on the sub-inference steps. If the competence-level is "intermediate" then the tutor only provides an inquiry for the inference so that the student can perform the inference by himself.

4.3. Learning Environment

AdvGeo tutor has several windows for students to learn how to make a proof. Figure 4 shows an imaginary screen shot of the tutor. Following subsections describe each window in details.

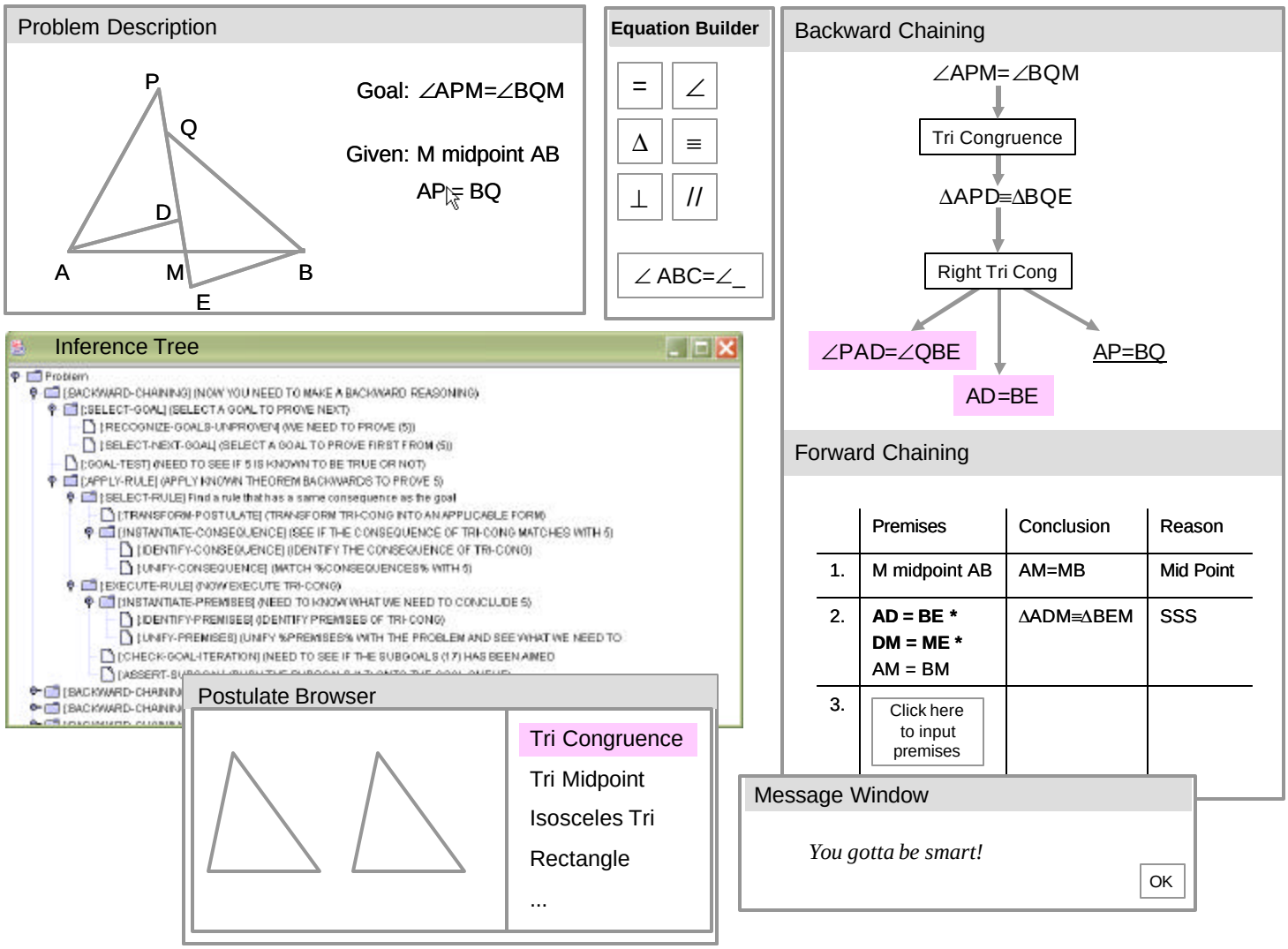

Figure 4: AdvGeo Graphic User Interface

Windows to provide message: Inference Tree, Messge, Postulate Browser

Windows to make a proof: Backward Chaining, Forward Chaining

4.3.1. Problem Description window

This window shows a problem that consists of the goal to prove, the givens, and the problem figure. The problem figure displayed in this window is used in two ways: (1) highlight elements appeared in a proposition used in elsewhere (e.g., the Backward and/or Forward Inference window), and (2) specify an element as a part of a proposition to be input (e.g., clicking segment AB followed by input = and a click on segment CD composes a proposition AB=CD).

4.3.2. Proposition Builder window

When students need to input a proposition, the Proposition Builder window automatically appears. This window has several buttons that represent mathematical symbols such that =, ∠, ≡, Δ, etc. To make a proposition, students must specify geometric elements and a relation. A relation can be input by typing in a keyboard or clicking one of the symbolic icons on the Proposition Builder window. Geometric elements can be input either by typing in a keyboard or clicking the corresponding elements displayed in the Problem Description window.

4.3.3. Postulate Browser window

The students can refer to the postulates they are supposed to utilize in a proof. Clicking a name listed in the Postulate Browser window shows a typical configuration for the postulate as well as its premises and a conclusion.

4.3.4. Inference Tree window

The Inference Tree window reifies the process of making a proof by showing a sequence of postulate application both forwards and backwards. A postulate application is further broke down into several inference steps. Since these inference steps are defined hierarchically, they are displayed as a tree that consists of nodes that has one or more branches, which are also nodes.

Since a proof has many postulate applications, each of which has several inference steps, a tree representing a proof is usually very many branches. Branches can be disappeared or appeared by clicking the node from which the branches diverge.

4.3.5. Backward Inference window

The Backward Inference window reifies process of backward chaining. The top-level goal is shown at the top most row of the window. A backward postulate application is shown as a box labeled by the postulate's name with two allows connecting its premises and a goal; incoming arrow from the top of the box connects

the postulate and its conclusion whereas the outgoing arrow(s) from the bottom connects the postulate and its premises. Propositions (i.e., premises) that are not supported are shaded. Propositions that are given in the problem is underlined (see AP=BQ in Figure 4).

Students are also supposed to use this window to make a backward inference. To make a backward inference, they first specify a proposition that is a consequence of the inference. This is done by clicking a proposition in the Backward Inference window. Once the consequence is specified, a box appears underneath in which the students are supposed to input the name of a postulate that support the inference. The students can either type the name in the keyboard or select the one shown in the Postulate Browser window. Finally, the students must input propositions as the premises of the inference. The Equation Builder automatically pops up at this moment.

There are two exceptional situations for backward chaining that do not follow above steps: (1) for premises that are given, students must use the word "Given" as the postulate name in the second step. In this case, the tutor does not ask students to input premises of the inference, and (2) when a premise input at the last step is "trivially true" (e.g., AB=AB), then it is automatically underlined hence no further justification is necessary.

4.3.6. Forward inference window

The Forward Inference window shows forward reasoning in a table. A row in the table shows single forward inference. The table column consists of Premises, Conclusion, and Reason. A row that has *a*, *b*, and *c* in the Premise, *x* in the Conclusion, and *p* in the Reason should be read "From *a*, *b*, and *c*, one can conclude *x* because of the postulate *p*."

Students can use the Forward Inference window to make a forward inference. They start from specifying premises, which is done by clicking an empty cell in the column Premise. The students then input premises of the inference by clicking existing propositions in the column Premise, or for the givens, by clicking propositions shown in the Problem Description window. They then input a proposition as the conclusion of the inference. The Equation Builder window automatically appears at this moment. Finally, the students must specify a postulate that supports the inference by either typing its name in the Reason column or selecting one in the Postulate Browser window.

4.3.7. Message window

When a student makes an erroneous forward or backward inference, then AdvGeo tutor provides a feedback on the error in the Message window. Students must close the Message window by clicking [OK] button.

5. Experiment Design

This section describes experiments to answer the research questions mentioned in Section 3.

5.1. Hypothesis

As a response to the research questions, we have hypothesized outcomes from the experiments described in the following sections.

- Comparing to the traditional intelligent tutoring system specifically a model tracing tutor and traditional classroom instruction, the AdvGeo tutor with the proactive scaffolding on thorough inference steps results in leading students to a certain level of problem-solving skills that can be assessed by the amount of near (or "within domain," if you will) transfer, namely, they would solve similar problems with a certain accuracy within a shorter amount of time.
- Comparing to a version of the AdvGeo tutor that does not provide proactive scaffolding on the "hidden" inference steps, the full version of the tutor leads students to a certain level of problem-solving skills in terms of the amount of time to reach a certain level of problem-solving skills measured by the amount of near transfer.
- Comparing to the traditional intelligent tutoring system specifically a model tracing tutor and traditional classroom instruction, the AdvGeo tutor that provides guided problem-solving environment leads students to a certain level of problem-solving skills in terms of the amount of time to reach a certain level of problem-solving skills measured by the amount of near transfer.
- The students appreciate to use the AdvGeo tutor where the appreciation is defined by the degree of liking for the tutor. (Must be analyzed qualitatively?)

5.2. Pre and Post Tests

For students' achievement in proof-writing skills with traditional classroom instruction, Senk (1989) reports intensive experimental study that utilizes the van Hiele geometry test and CDASSG proof test. She showed that the van Hiele level is known to be a good predictor of student's achievement in proof skills (citation). Hence we also utilize the van Hiele geometry test as a pretest and CDASSG proof test as a post test.

For comparison with the model tracing tutor, we utilize the proof tests, invented by Koedinger (1991), that has been used to asses students improvement in proof skills. The proof tests are grouped to form the Test A and the Test B. Like the study conducted by Koedinger, we use Test A and B both for pre- and post-test to counter-balance the influence of the pre-test to the performance in the post-test.

We must merge CDASSG proof test and Koedinger's proof test so that students only take a single pre- and post-test.

5.3. Analysis

6. Expected Contribution to the ITS Community

7. Milestone

Reference:

EN.REFLIST