**International Journal of Advanced Educational Research**
**ISSN: 2455-6157; Impact Factor: RJIF 5.12**
**Received: 13-09-2020; Accepted: 28-09-2020; Published: 14-10-2020**
**www.educationjournal.org**
**Volume 5; Issue 5; 2020; Page No. 26-30**

# An intelligent tutoring system-style assessment software that diagnoses the underlying causes of students' mathematical mistakes

**Ivy Liang[1], John Leddo[2]**
[1] Director of Research, EdMaster LLC, Sunrise Valley Drive, Herndon, Virginia, United States
[2] Researcher, EdMaster LLC, Sunrise Valley Drive, Herndon, Virginia, United States

**Abstract**
Traditional large-scale and high-stakes assessments have focused largely on whether test takers give the correct or incorrect answers to questions. Early instructional software followed this paradigm. The introduction of intelligent tutoring systems (ITSs) led to an emphasis on discovering where students were making mistakes and explaining the mistakes through matching them to pre-defined error catalogs. The deficiencies with this approach were an emphasis on identifying only procedural mistakes and not validating whether the matched errors were, in fact, the true causes of mistakes. The present paper describes an ITS-style assessment software that diagnoses causes of errors by assessing underlying and pre-requisite concepts a student needs to solve a problem. The assessments focus on a variety of knowledge types: abstract concepts, procedures, and ability to apply concepts to problems. The software even assesses whether a mistake was caused by carelessness or mistyping information from the problem. The software was evaluated by comparing its agreement in diagnosing causes of students' mistakes with that of experienced teachers. Results showed that the software's agreement percentage was in the 90s and statistically equal to that of experienced teachers' inter-rater agreement.

**Keywords:** intelligent, Underlying, mathematical, statistically

## Introduction

Assessment has been an integral part of the educational process since its inception. Assessment, particularly when large-scale or high-stakes, has traditionally focused on measuring how many correct and incorrect answers test takers give. Examples of this include standardized tests such as the SAT or state assessments such as Virginia's (US) Standards of Learning.

It is only natural, then, that instructional software such as computer-based training (CBT) or computer-based instruction (CBI) followed this assessment paradigm. Assessment would identify the problems a student missed and then remediate instruction was given on the topic(s) the missed problem(s) were associated with. A good example of this is the software on the highly popular Khan Academy website.

The drawback with this type of assessment is that it focuses on *whether* a mistake is made but not *why* the mistake is made. This type of deficiency in the assessment and instructional process inspired the next generation of instructional software, intelligent tutoring systems (cf., Brna, Ohlsson and Pain, 1993 [1]; Greer, 1995) [1]. An intelligent tutoring system (ITS) is an artificial-intelligence-based instructional system that uses a knowledge model (called an expert model) of the mastery knowledge to be taught the student, a cognitively-based pedagogical model for teaching that knowledge and a student model for tracking student progress toward that mastery knowledge.

In a prototypical ITS such as the Cognitive Tutor (Koedinger and Corbett, 2006) [4], the system contains a problem-solving protocol that the student is taught and expected to follow. If the student strays from this process, the type of mistake made is matched to an error catalog. Once the type of error is identified, remediation is presented to the student based on the error type. This matching of error to error catalog may very well be the first paradigm for trying to identify *why* an error occurred rather than *that* an error occurred. The benefits of this approach were immediate: while tradition CBT produced average educational gains of. 38 standard deviations, ITS produced average educational gains of. 66 standard deviation (Kulik and Fletcher, 2016) [5].

While the use of problem solving protocol and error catalogs represented a step forward in identifying more precisely where a student made his or her mistake when problem solving and pointed to a possible explanation for why the mistake was made, a weakness of this approach was the lack of validation of whether the cognitive deficiency actually identified in the error catalog was actually present in the student. A second weakness of this approach is its limitation in identifying what type of knowledge deficiency could result in the observed mistake.

Given that the cognitive psychology literature has identified that people possess different types of knowledge such as semantic (Quillian, 1966) [11], procedural (Newell and Simon, 1972) [10], schematic (Schank and Abelson, 1977) [12] and causal (de Kleer and Brown, 1981) [2], it seems limiting for assessment software to restrict its diagnosis of errors to procedures a person failed to execute correctly. The above literature suggests that mistakes could result from a variety of knowledge deficiencies or applications of knowledge to problem solving. Assessment instruments should have the ability to diagnose what type of knowledge or application thereof is responsible for problem solving mistakes.

John Leddo has attempted to address this weakness in assessment technology. First, he developed, in conjunction

with Marvin Cohen, an assessment methodology called Cognitive Structure Analysis (1989) or CSA. CSA had two main components: a knowledge model that consisted of the different types of knowledge needed for problem solving and a set of probes designed to determine which of these types of knowledge a person had.

CSA was next implemented in a software format (Leddo and Sak, 1994). Students were probed before and after instruction to determine how much they knew about a subject area (in this case, scientific problem solving). Leddo and Sak found that the knowledge models built using the automated CSA process correlated. 88 with student problem solving performance and the change in assessed knowledge using CSA correlated. 78 with change in problem-solving performance.

Leddo next extended this research to compare the effectiveness of the CSA-based approach in assessing student knowledge to that of trained teachers. To accomplish this, Leddo *et al*. (1999) created a virtual biology lab in which students conducted a simulated experiment, recorded their data and analyzed it on a virtual worksheet and then responded to queries to assess how much they understood the underlying concepts behind the experiment they were conducted. Leddo *et al*. found that, when identifying student learning needs and performance against state standards, the software they created agreed with assessments produced by experienced teachers 97% of the time which was statistically equal to the rate with which teachers agreed with each other. While this assessment software examined a variety of knowledge types and even integrated different forms of input (behaviors on a simulator, a workproduct created in the form of a datasheet, and responses to queries), it did not probe students on specific mistakes made to diagnose the underlying causes.

The next major iteration by Leddo and his colleagues (Leddo *et al*., 2019) involved the incorporation of voice and natural language processing into the assessment process. In this work, students are queried using voice technology and they give their responses orally. The queries are based on CSA and any missed concept is immediately remediated.

The work of Leddo and his colleagues is noteworthy since it attempts to link underlying concepts to problem solving performance and compare the effectiveness of the software to that of experienced teachers. The next step, which is the focus of the present research, is to link specific mistakes made by students to underlying knowledge deficiencies, so that remediation can be targeted to the specific concept that needs to be remediated.

In order to accomplish this goal, three considerations need to be taken into account. First, our years of experience in tutoring at MyEdMaster reveal that often a mistake is not caused by a weakness in the subject matter that the presented problems are based on but by a weakness in pre-requisite concepts that the student has grown rusty on or perhaps was never taught in the first place. Therefore, knowledge related to both the current topic and pre-requisite topics both have to be assessed.

Second, students often understand a concept in principle, but have trouble applying the concept to actual problems. While this does represent an actual weakness, the implication is not simply that the student does not know the concept. Therefore, the remediation needed may be different than simply having the student relearn the material or point out the correct way to do the step (s) that the student missed.

Third, our experience also teaches us the commonsense observation that students are not perfect. Often, they have the concepts needed to solve a problem, but they make careless mistakes. Careless mistakes can take many forms. The two most common ones that we have observed in students are arithmetic computational errors and mistakes in copying values from the problem. The implication for these mistakes is that students may not need remediation for any particular topic but rather additional practice in computations or reminders to read the problem more carefully.

The remainder of the paper describes an ITS-style software that assesses students who solve two-step algebra equations. The software performs a two-pass assessment process, first looking at the process a student goes through and then diagnosing any underlying knowledge deficiencies that may be responsible for errors that students make. The software tries to determine whether any of the mistakes could also be the result of a student's failure to apply a concept to an actual problem as well as a careless mistake. The software is then evaluated by comparing its assessment of student knowledge to those of two experienced math teachers.

**Assessment Software**

The goal of the assessment software is twofold: to assess whether a student can follow a correct solution path when solving a math problem and, if the student makes an error, to diagnose the cause of any mistakes the student makes. Written in Python, the software utilizes a two-step process. First, a student is given a two-step equation problem of the form ax + b = c to solve. The values of a, b and c are integers that are randomly generated by the software. The student then enters his or her problem-solving process directly into the software. The software has an underlying protocol, which corresponds to the expert model in traditional ITSs, that forms the basis of a correct solution (which matches how the student is taught how to solve this type of problem—see Method section below). The protocol has 4 steps:

Step 1: ax + b - b = c - b
Step 2: ax = c - b
Step 3: ax/a = (c - b)/a
Step 4: x = (c - b)/a

Within the software's knowledge model is a list of prerequisite concepts that a person must know to be able to solve two-step equations successfully. The prerequisite concepts are associated with each step of the problem-solving process. These concepts are listed by step below.

1. The list for step 1 has "isolating the variable," "using the additive inverse," and "balancing the equation."
2. In step 2, the only conceptual error the student can make is "arithmetic error."
3. In step 3, the prerequisite knowledge is "multiplicative inverse."
4. In step 4, the only conceptual error the student can make is "arithmetic error."

Once the student goes through the problem-solving process, the software identifies steps in which errors were made. The software then goes through the problem steps that were missed and assesses the underlying concepts. In order to discuss how this is done, we first outline the types of conceptual errors the software tests for. A total of eight

possible conceptual errors exist: isolate the variable, additive inverse, balance the equation, arithmetic error, multiplicative inverse, application misunderstanding, carelessness, mistype.

In order to discuss how the software determines which conceptual error(s) exist in the student, we start by defining each concept and then say how the software assesses it.

Isolate the variable refers to creating a problem form where the x exists all by itself on one side of the equation and its value exists on the other. Isolating the variable involves removing the non-variables from the left side of the equation. If the student fails to do so and demonstrates a lack of knowledge of how to isolate the variable, then the software concludes that the student does not understand the concept of isolating the variable. If the student makes a mistake on the first step, the software asks the student to list the generic form of the first step, which is ax + b -b = c-b. If the student does not demonstrate an understanding that the value b needs to be eliminated from the left side of the equation (e.g., by performing a subtraction from the left side), then the software concludes that the student lacks knowledge of isolating the variable.

If the student demonstrates that s/he knows that the variable needs to be isolated, the software next evaluates whether the student understands the concept of additive inverse, which is a pre-requisite concept. If the student's input for the first step of the problem indicates a subtraction on the left side, the software checks to see if the number subtracted has a value of -b. If so, the student is deemed to understand the concept generally. The student would then be asked to redo the step in the problem. If the student gets it right, then the student was assumed to have made a careless error. If not, then the student was assumed to have made an application error, namely, s/he has the general concept but makes a mistake applying to a specific problem. The student did not indicate that s/he would subtract -b from the left side of the equation, the software assesses that the student is lacking the concept of additive inverse.

If the student does subtract -b or its actual value in the actual problem from the left side of the equation, the software checks to see if the student subtracted the same value from the right side of the equation. If the number is different, then the software tries to determine whether there is a balancing the equation deficiency. If the student indicates in the probing that s/he would subtract -b and its corresponding value from the right side of the equation, then there is no error. If the student does not indicate this, then the system evaluates the generic case in the follow-on probing as described above followed by application to the specific example. If the student, misses the general case that -b needs to be subtracted from both sides of the equation, then this is a balancing the equation error. If the student gets the general case right but makes a mistake in the specific problem, then this is an application error. If the student gets both the general case and the specific question right in the probing, then this is a careless error.

In the second step of the problem, the only possible mistake is an arithmetic mistake. Here, the follow-on probing determines whether there is a need or whether the student made a careless error.

In the third step of the problem, the pre-requisite knowledge is multiplicative inverse. The process on this step is a lot like that for additive inverse. For the fourth step, the only possible mistake is an arithmetic error when simplifying (c-b)/a. This assessment is similar to that in the second step. The only other possible error is called mistype. This occurs when a student miscopies information from the problem, such as writing down the wrong coefficient of the variable when doing one or more steps of the problem.

One of the things that makes the present software unique is that it assesses abstract knowledge and application of concepts as well as procedural knowledge. The software assesses when students lack abstract concepts such as isolate the variable or balance the equation, concepts that are used in a variety of problems across a variety of topics. This stands in contrast to software products such as the Cognitive Tutor that focus exclusively on procedural mistakes.

Similarly, the software assesses whether students understand the abstract or parameterized version of each procedure. For example, not only does the software assess whether a student can find the right additive inverse for a specific problem, but whether the student understands the general process of finding an additive inverse when solving an equation. In other words, the software assesses whether students understand the general steps when solving a specific problem as well as the ability to solve the specific problem itself. This enables the software to assess application errors, i.e., knowledge of the general concept but an inability to apply the concept to a specific problem. The implications for remediation are important, since the emphasis of such remediate would be on general application of a formula, not finding the right step for a specific problem.

## Method

### Participants

Participants were ten middle school students recruited from schools in Montgomery County, Maryland in the United States. Participants did not have prior knowledge of solving two-step equations. There were two middle school teachers who taught pre-algebra and algebra 1 and who served as the teachers providing assessments of students' problem-solving knowledge.

### Materials

Each student solved one problem that was generated by the software being evaluated. The problem was always given in the form ax + b = c, a generic two-step equation format. The values for a, b and c were randomly generated non-zero integers.

### Procedure

First, the ten students were given a sample two-step equation question to verify that they did not know how to solve these types of problems. Then, students were taught how to solve these types of problems and also how to use the present software. The students were then given a single problem created by the software and asked to solve the problem and type their work directly into the program. In cases where they made mistakes, they responded to any probes presented by the software by typing their responses directly into the software.

While the students went through the process of completing the problems, their steps were recorded by a screen recording program on the computer. Then, both teachers went through all ten recordings, analyzing the students' responses. They were provided a list of all pre-requisite concepts/potential error types to look for. This was done to

insure common terminology so that comparisons could be made between teachers and between teachers and the software. After each video, the teachers were requested to write down what errors, if any, the students made. Each teacher worked separately and had no knowledge of what errors the other teacher or the software was listing in their assessments of student work.

**Results**

The main research question being investigated is whether the software's assessment of student performance and diagnosis of underlying causes of mistakes would match those of experienced teachers. As discussed above, the two-step equation problem solving process involved eight possible concepts/mistakes: isolate the variable, additive inverse, balance the equation, arithmetic error, multiplicative inverse, application misunderstanding, carelessness, mistype.

Given that there were 10 participants in the study, a total of 80 judgments were made by each teacher and the software as to what concepts were known or mistakes were associated with each participant. Table 1 shows the agreement rate between teachers and teachers and software, by student, for each of the nine assessed concepts/mistakes.

**Table 1:** Interrater agreement between teachers and teachers and software

| | **Teacher 1 vs. Teacher 2** | **Teacher 1 vs. software** | **Teacher 2 vs. software** |
|---|---|---|---|
| Student 1 | 8 of 8 | 7 of 8 | 7 of 8 |
| Student 2 | 7 of 8 | 7 of 8 | 6 of 8 |
| Student 3 | 7 of 8 | 6 of 8 | 7 of 8 |
| Student 4 | 8 of 8 | 8 of 8 | 8 of 8 |
| Student 5 | 8 of 8 | 8 of 8 | 8 of 8 |
| Student 6 | 7 of 8 | 7 of 8 | 8 of 8 |
| Student 7 | 8 of 8 | 8 of 8 | 8 of 8 |
| Student 8 | 8 of 8 | 7 of 8 | 7 of 8 |
| Student 9 | 7 of 8 | 7 of 8 | 8 of 8 |
| Student 10 | 8 of 8 | 8 of 8 | 8 of 8 |
| Total | 76 of 80 | 73 of 80 | 75 of 80 |

Overall, the interrater agreement rate between teachers is. 95 and the interrater agreement rate between teachers (combined) and the software was. 93. This difference is not statistically significant, Z =. 67, ns. While disagreements between teachers and even between teachers and software were rare, it pays to take a closer look at the nature of those disagreements.

The present software may be relatively unique in that if a student makes a mistake, the software assesses the underlying concept to see if the student understands that concept and can correct the mistake. If the student has the concept, then the software assumes that the students made a careless mistake. Similarly, the software catches mistyped information from the problem. Arguably, other than further practice and guidance to be more careful, no real concept remediation is required for mistyped information or careless mistakes. However, the identification of mistypes and careless mistakes may be useful in avoiding spending time remediating concepts that are understood, although the opportunity cost of reviewing concepts that students do understand is probably small. Accordingly, the most useful diagnostic information the software can provide is to identify genuine gaps in underlying concepts that are needed to master a topic.

In order to understand how well the software can diagnose underlying conceptual learning needs, a second analysis was done. In this case, a comparison of teacher vs. teacher and teacher vs. software was done on assessed concepts, excluding careless mistakes and mistypes. This meant that there were six concepts to be assessed for the two-step problem solving: isolate the variable, additive inverse, balance the equation, arithmetic error, multiplicative inverse, application misunderstanding.

When teacher and software assessments are examined for these six concepts, a different pattern emerges. Teachers and software had complete agreement on their assessments for eight of the ten students. For the two students, each teacher disagreed with each other on one skill per student and the software disagreed with each teacher on one skill per student (although, interestingly, there was no overlap in concepts on these three disagreements). Overall, there was agreement on 58 of 60 assessed concepts or an interrater agreement of. 97 for both teacher-teacher and teacher-software. These results are comparable to those found in Leddo *et al*. (1999).

**Discussion**

The present study found that an ITS-style was able to diagnose the causes of student errors with the same reliability as human teachers. Across the board, the software's assessment of the student agreed with those of experienced teachers 93% of the time, which was statistically equal to the 95% inter-rater agreement between teachers. When the assessing only actual topic-based knowledge, the software agreed with experienced teachers 97% of the time, equal to the 97% inter-rater agreement between teachers.

The main innovation here is that once the software detected that a student made a problem-solving mistake, it initiated a querying process in order to identify what missing concepts were responsible for the students' mistake. Unlike other ITS software that may simply identify at what step a mistake occurred and what procedure was not followed, the software evaluated a mistake at several different levels: conceptual understanding, procedural, ability to apply concepts to problems and even careless mistakes. This is an important innovation. As discussed in the Introduction people have different types of knowledge and assessment instruments should be able to diagnose which type of knowledge is responsible for mistakes.

The value of this type of assessment technology lies not only in the ability to diagnose why people make mistakes but also in how best to remediate them. Remediation of abstract knowledge may be best accomplished by focusing on the rationales behind the concepts involved (e.g., why it is necessary to balance an equation). Remediation of application of knowledge errors may be best accomplished by focusing on teaching students how to extract information from problems or what instantiations of abstract principles look like in actual problems. Remediation of procedural errors may be best accomplished through repetition of problem solving. Remediation of careless errors may be best accomplished by having students check their work.

The present study focused on creating and validating assessment technology that diagnoses the causes of student errors. Future research will link this technology to the remedial process to show how diagnosis of the causes of

student errors can lead to improved remediation, resulting in improved student problem solving.

**References**


1. Brna P, Ohlsson S, Pain H. (Eds.). *Proceeding of Artificial Intelligent in Education '93.* Charlottesville, VA: Association for the Advancement of Computing in Education, 1993.
2. de Kleer J, Brown JS. Mental models of physical mechanisms and their acquisition. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Erlbaum, 1981.
3. Greer J. (Ed.) *Proceeding of Artificial Intelligent in Education '95.* Charlottesville, VA: Association for the Advancement of Computing in Education, 1995.
4. Koedinger KR, Corbett AT. Cognitive Tutors: Technology bringing learning science to the classroom. In K. Sawyer (Ed.) *The Cambridge Handbook of the Learning Sciences*, Cambridge University Press, 2016, 61-78.
5. Kulik JA, Fletcher JD. Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review. *Review of Educational Research. 2016; 86(1):*42-78.
6. Leddo, J. and Cohen, M.S. Cognitive structure analysis: A technique for eliciting the content and structure of expert knowledge. Proceedings of 1989 AI Systems in Government Conference. McLean, VA: The MITRE Corporation, 1989.
7. Leddo J, Guo Y, Liang Y, Joshi R, Liang I, Guo W, *et al*. Artificial intelligence and voice-powered electronic textbooks. International Journal of Advanced Educational Research. 2019; 4(6):44-49.
8. Leddo, J. and Sak, S. Knowledge Assessment: Diagnosing what students really know. Presented at Society for Technology and Teacher Education, March, 1994.
9. Leddo, J., Zhang, Z., Huang, J., Kalina, P. An Internet-deliverable Automated Performance Assessment Tool. (Technical Report 99-2) Herndon, VA: Research Development Corporation, July, 1999.
10. Newell A, Simon HA. *Human problem solving.* Englewood Cliffs, NJ: Prentice Hall, 1972.
11. Quillian MR. *Semantic memory*. Camridge, MA: Bolt, Beranek and Newman, 1966.
12. Schank RC, Abelson RP. *Scripts, Plans, Goals, and Understanding.* Hillsdale, NJ: Erlbaum, 1977.