Using PADI to Develop an Online Self-Assessment System that Supports Student Science Learning in FOSS

Cathleen Kennedy, University of California at Berkeley
Michael Timms, University of California at Berkeley
Kathy Long, Lawrence Hall of Science
Susan Ketchner, Lawrence Hall of Science
Dan Bluestein, Lawrence Hall of Science

Presented at the American Educational Research Association annual conference,
April 14, 2005
Montreal, Canada

This work is supported by the National Science Foundation under grant REC-0129331 (PADI Implementation Grant). Opinions expressed are those of the authors and not necessarily NSF. Please do not cite or quote without permission of the authors.

Background

The FOSS (Full Option Science System) Project at the Lawrence Hall of Science has recently developed a middle school course called Force and Motion. The PADI/FOSS team has developed an online, interactive Self-Assessment System to reinforce students' classroom learning. The system provides an instructional mode, in which students practice solving problems, and an assessment mode that provides feedback about progress. The practice mode uses student proficiencies to determine the type of hint the student is most likely to need, while the assessment mode uses proficiencies to provide feedback to students and teachers about progress. The practice mode functions as a tutorial until students feel ready to test themselves by requesting a "Quick Check" to see their progress. In this paper we discuss how the PADI assessment tools provided support for the development and implementation of the Self-Assessment System.

Conceptual foundations of the Self-Assessment System

Knowing when a student needs help and what help to give is at the heart of effective tutoring. Making that determination is especially challenging for a computer-based interactive learning environment (ILE) because it involves tracking information about the student's performance in a format that can be interpreted by the computer system. Human tutors may use verbal and visual cues, like hesitancy in speech or body language, but ILEs must rely on other cues. The approach to tutoring that is embodied in the Self Assessment system described in this paper is derived from a review of a number of empirical studies of human tutoring and computer-based tutoring. From this review we developed a set of characteristics of effective tutoring that are useful in designing a computer-based system that can adapt to students' needs. The characteristics of effective human tutoring are summarized below.

> **Tutors lead the tutoring process –** Katz et al (2003) found that human tutors initiated two thirds of the tutoring dialogues after a student completed a problem-solving episode, while students only initiated one-third. However, although tutors more often initiated the tutoring process, students preferred to be led to a

productive learning path, rather than be forced to follow it (Lewis et al, 2003). For example, a learner might be given feedback that he or she is using the wrong method in solving a problem and allowed to pick another method, rather than the tutor imposing the correct method.

Good tutors are able to adapt – Human tutors are not always successful on their first attempt at helping a student with a hint. However, they are adept at recovering from a failed tutorial interaction, and are able to quickly rephrase or explain in a different way (Johnson, 2003). In an early study of mothers tutoring their 3-4 year olds, Wood and Middleton (1975) found that learning outcomes for tutees varied by the specific features of the tutorial activity. For example, they found that if the tutor made a verbal suggestion with which the child did not comply, the tutor provided more help. However, if the child followed a suggestion, the tutor would allow the child to progress with more scope for autonomy by simply acknowledging the child's progress, or remaining silent. The process in which the tutorial interventions reflect the moment-to-moment activity of the child as he solves problems has become known as "instructional contingency"

Successful tutors adopt a conversational style – Not surprisingly, considering we are highly social animals, humans prefer tutors to use a conversational style. Tutors often use phrases that incorporate the term "we." For example, " Why don't we go back to the previous step of the problem." Sometimes they express their feedback using the first person. For example, "I would start at the top." (Johnson, 2003). Verbal feedback can also lighten the cognitive load when the student is already engaged in the practice or problem solving process (Roberts, Pioch and Ferguson, 2000).

Effective tutors use hints – Tutors often offer hints (Johnson 2003) as the opening to their tutorial offering. Hints are offered in a variety of ways, often as a suggestion or as a question.

Hints are not the only tutoring strategy – Human tutors use not only hints, but also questioning and dialogue in a Socratic style, allowing students to follow an incorrect path. This allows students to get stuck and, hopefully, realize their error. The tutor simply flags the errors for later discussion.

Tutors structure their feedback – In developing the verbal tutor for the computerized TRANSoM system, which coaches students who are learning to control remotely operated vehicles, Roberts et al (2000) found it beneficial to give a student just enough feedback to allow him or her to make a correction, without giving too much away. In situations where students were making a number of simultaneous errors, the system was designed to prioritize which error was more significant and to give feedback on that one only to prevent the student from being swamped in a cacophony of tutoring responses. TRANSoM's feedback becomes increasingly specific the longer the errors are ignored.

Timing of feedback to students is important – Tutors tend to give different feedback at different times in the problem solving process. During problem solving is the time for advice about what action to take next or feedback about the relationship of a particular action to a global strategy for solving a type of problem. Tutors commonly leave tactical or strategic feedback on how to approach a particular problem or how to classify problems until the student has completed their solution. Similarly, general feedback like tips that might be considered "tricks of the trade" are left until the end of the problem solving, as is feedback of a conceptual nature. Tutors avoid interrupting a student's flow of thought in the problem-solving process with such feedback. In other words, feedback that might be thought of as metacognitive seems most effective at the end of a problem-solving episode.

Post-solution feedback is important – Katz et al (2003) studied the effectiveness of tutoring feedback during problem solving and after problem solving (post-solution). They found that post-solution feedback and reflective questions in combination with feedback during problem solving was more effective than feedback only during problem solving. In particular, post-solution dialogues supported learning when they were initiated by the tutor, focused on abstract concepts or strategies associated with the current problem, and elaborated upon or restated discussion from the problem-solving stages.

> **No difference between human feedback and "canned" feedback –** An encouraging finding for developers of computer-based help systems was made by Katz et al (2003). There were no statistically significant differences in learning gains between students who received human feedback as part of their post-solution tutoring and those who received "canned" feedback provided by the computer system.

The great hope of interactive learning environments is that the human tutoring process can be analyzed, reduced to algorithms, and implemented in computer-based tutoring. Then, ILEs would be able to extend to many students the opportunity of one-to-one tutoring that is currently only a pipe-dream in our increasingly large classrooms of 30 or more students. Unfortunately, human tutoring has proved to be a complex activity that has not been studied as extensively as is needed to be able to build demonstrably effective tutoring strategies and algorithms for ILEs (Aleven et al, 2003).

Although almost all interactive learning environments provide help to students, there have been few studies that focus on how such help should be tailored to a student's particular ability level and needs. In the early days of developing ILEs, there were many other challenges to meet, such as tracking actions of the learners and modeling their learning; it may be that the process of providing help seems fairly simplistic and has therefore not received the attention it deserves. A review of the studies that have examined providing help in intelligent learning environments reveals that the provision of help is, in fact, a complex topic.

Student help systems fall broadly into two main categories: on-demand help and automated help. On-demand help systems are ones in which the learner controls when they seek help, and which help to choose. Automated help systems are ones in which the learning system controls the provision of help, and assistance is delivered when the system judges that the student needs it and in the format that the system determines is optimal. Both systems have advantages and disadvantages.

The best summary of the state of research on on-demand help in interactive learning environments is provided by Aleven et al (2003). In their conclusions, Aleven et al point out that one of the main issues in providing on-demand help is that learners often fail to use such systems effectively and may even ignore them completely. Three separate studies have shown that students tend not to use on-demand help offered in ILEs. Mandl, Graeser and Fischer (2000) report on a series of studies on problem-oriented learning in which medical students solved diagnostic cases on their own. They found that even advanced medical students made inadequate use of the help offered in the system. The ILE presented cases on anemia and related diseases for the students to diagnose, and offered several types of help to students, including a glossary of medical concepts, a link to lectures containing relevant background knowledge, and automated advice from an expert diagnostician. Mandl et al found that, despite these available help offerings, students tended to try to solve the diagnostic cases unaided. Think-aloud protocols revealed that even after gaps and failures were revealed to students, they did not seek help.

Another quasi-experimental study cited by Aleven et al (2003) conducted by Hofer, Niegemann, Eckert and Rinn (1996) found that even when students report that a help system provides useful direction, they still might not use it frequently. Hofer et al found that students value a help system more highly when it offers help that is directly related to the task to be performed. However, even then, students seldom used it. In a second study, Hofer compared students' use of two different types of help, one in which the system intervened with help when the student made a mistake and another which was available on demand at various points and offered a hyper-linked text book. Students judged the problem feedback to be more useful than the hyperlinked text, and rarely used the text.

In the third study showing ineffective use of on-demand help, Aleven and Koedinger (2000) examined help seeking in a version of the Geometry Cognitive Tutor. The Geometry Tutor provides context-sensitive hints on demand, which are specific to the current problem and to student progress on the problems up to that point. Hints are presented in levels (up to eight) that give progressively deeper help that reveals more and more of the correct solution until the answer is given to the student. The system also provides an on-line glossary that lists relevant problem solving skills and provides examples. Aleven and Koedinger found that students tended to quickly click through the hints to reach the bottom level hint, which provides the deepest level of help, and spent little time with hints that explained why the answer was the way it was. Students also largely ignored the extensive glossary help. Such help-seeking behaviors were deemed unlikely to promote learning. In conclusion, on-demand help can be helpful, but tends to be used so ineffectively that it is not supportive of learning.

The other type of help system is automated help, in which the system identifies when a student needs help and judges the type of help to deliver at the appropriate time. There is little literature dealing with this precise topic. In their 2003 paper entitled *Help Seeking and Help Design in Interactive Learning Environments*, Aleven, Stahl, Schworm, Fischer and Wallace provide an excellent review of what is a fairly scant body of research. Aleven et al cite studies that have begun to reveal the characteristics of providing help in Interactive Learning Environments (ILEs) that appear to be effective and under what circumstances.

Below is a summary of the main findings about the most effective types of help to provide in interactive learning systems.

> **Types of help –** Function-oriented, or principle-oriented, help should be the primary method of help in ILEs, while operative help should be offered only in

the last resort. Dutke and Reimer (2000) found that operative help (help with steps along the way to a specific goal) was effective in supporting performance throughout the task at hand, but principle-oriented help (explanations of how a particular principle works) was more effective in producing learning that could be transferred beyond the immediate task.

Types of hint – The type of hint that is most beneficial depends on the learner's cognitive ability. In two studies of a mathematics ILE called Animalwatch, Arroyo and her colleagues (Arroyo, Beck, Beal, Wing, & Woolf, 2001; Arroyo, Beck, Woolf, Beal & Schultz, 2000) compared two types of hint. C*oncrete hints* were ones that made reference to concrete objects such as base-10 blocks and bars, and made connections to real-life problems. *Abstract hints* used direct operations over numerals. Concrete hints were found to be more effective with students with a low level of cognitive development, whereas abstract hints were more useful to students at higher levels of cognitive development. This suggests that to achieve maximum impact, the level of abstraction of the hints provided to learners should be matched to their level of cognitive development.

Interactivity of hints – Arroyo et al (2000, 2001) found that learner self-confidence and performance on tasks were affected by the level of interactivity of the hints, and were related to gender. They compared hints with a highly interactive multimedia component in which students were asked for several kinds of input with ones that had short messages and in which students only had to provide minimal input. Boys had higher self-confidence and performance when using the shorter, less interactive hints. When the hints were more interactive, boys' self-confidence declined. Girls' self-confidence was unaffected by the level of interactivity, but they performed better on tasks when supported by more interactive help.

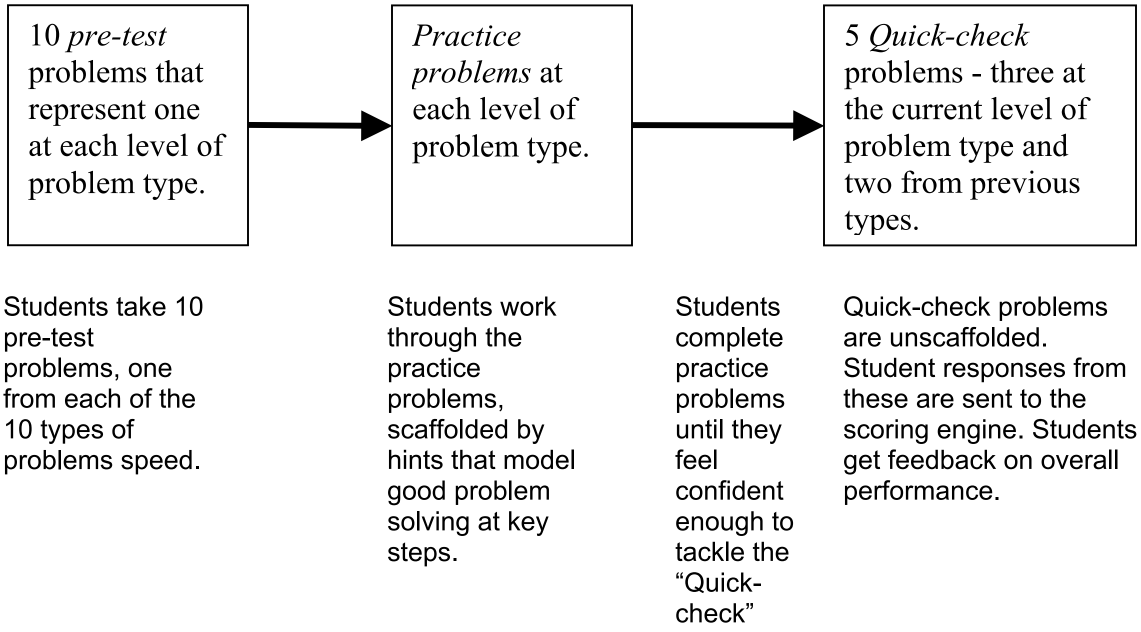
Type of information in the feedback – McKendree (1990) investigated the use of four different types of feedback information in a tutor for teaching geometry

> proofs. One type was *minimal feedback* in which students were notified that they had made an error, but gave no information on what the error was. The second type was *condition violation feedback* in which students were told that the geometry rule they had selected could not be applied as intended. The third type was *goal feedback* that stated the correct sub-goal on which the student should focus. The fourth type of feedback was *combined feedback* that stated both the subgoal and an unsatisfied rule condition. McKendree found that goal feedback produced better learning than minimal feedback on the error or its cause (the condition violation feedback). Goal feedback led to better knowledge transfer to tasks that were unsupported by feedback, and learners were more likely to correct their errors after goal feedback than after the other types of feedback.

It is evident that there is a body of knowledge about what makes for effective feedback, but to date, few intelligent learning systems have systematically taken account of these findings. It seems probable that an automated help system that mirrored, as far as possible within the limitations of the computer system, effective practices from human and computer tutoring would be more effective than one that lacked such a theoretical basis. This paper describes the development of a Self-Assessment System for middle-school students that are an interactive learning environment for scaffolding student learning about distance and speed. It was designed to embody as many of the principles of effective human and computer tutoring as possible within the constraints of the delivery system.

General description of the FOSS Self-Assessment System

The Full Option Science System (FOSS) curriculum that included units on distance, speed and acceleration was published in a second edition in January 2005. During the final stages of its development, curriculum developers at the Lawrence Hall of Science turned their attention to creating assessment items that could form the basis of a student self-assessment that was to accompany the revised curriculum unit. A Self-Assessment System was developed in which students received feedback while they practiced on problems and were also able to monitor their own progress through short summative assessments. The development was part of the work on the NSF-funded Principled Assessment Designs for Inquiry (PADI) project that also involves the University of Maryland, SRI International, and the University of California, Berkeley.



10 *pre-test* problems that represent one at each level of problem type.

Practice problems at each level of problem type.

5 *Quick-check* problems - three at the current level of problem type and two from previous types.

Students take 10 pre-test problems, one from each of the 10 types of problems speed.

Students work through the practice problems, scaffolded by hints that model good problem solving at key steps.

Students complete practice problems until they feel confident enough to tackle the "Quick-check"

Quick-check problems are unscaffolded. Student responses from these are sent to the scoring engine. Students get feedback on overall performance.

Figure 1: Broad design of the Self-Assessment System.

The purpose of the self-assessment is to allow students, as they reach the appropriate part of the FOSS curriculum, to practice on problems like the ones in the curriculum unit that they are being taught. As shown in Figure 1, the self-assessment has three parts: a short *pretest*, the *practice problems,* and the *quick check*. Students first take a *pretest* so the system can determine their initial ability level. There are ten different levels of problems in the curriculum and students answer one problem from each of the ten levels in the pretest. When working through *practice problems,* students gain experience with problems that are similar to those that they recently encountered in the classroom with their teacher. In this phase of the system, students get support from the computer system in the form of feedback on their responses and hints to aid their progress. Students work on problems at one level until they feel confident to go on, at which point they take a quick check. In the *quick check*, students complete a set of up to five problems that include three from the current level and one each from the previous two levels. Based on the score from the quick check, the system makes decisions about whether the student needs more practice on problems at that level or is ready to proceed to the next level.

Use of the PADI System in the Design of the Assessment Program

As stated above, the Self-Assessment System has ten levels of problems representing successively more difficult problem solving contexts. Designers wanted students to see different problems at each level throughout the practice sessions and in the Quick Check. To accomplish this, the system was designed to use item shells in which the main body of the question remained constant, but the numeric values in the problems were randomly selected from an appropriate (to the specific problem) range of values. The item shells for each problem level were developed by using the PADI Design System, and stored as task specifications within the PADI database.

In the current project, development of the task specifications began with an analysis of what was to be measured by the assessment overall. Designers discussed trade-offs between detailed individual measures on each of the cognitive components involved and an aggregated measure of knowledge. The individual measures we considered included knowledge of distance concepts, knowledge of speed concepts, knowledge of acceleration concepts, and mathematical ability. Ultimately, the decision was made to produce one content-oriented measure and one mathematical ability measure from the assessment. This measurement objective of the assessment, termed a Student Model in the PADI Design System, included two student model variables, one for knowledge about distance, speed and acceleration (DSA) and one for mathematical ability (Math). The Student Model "DSA + Math" became the basis for all of the task specifications; the tasks that students eventually interact with were intended to elicit evidence that would indicate levels of knowledge on DSA and Math.

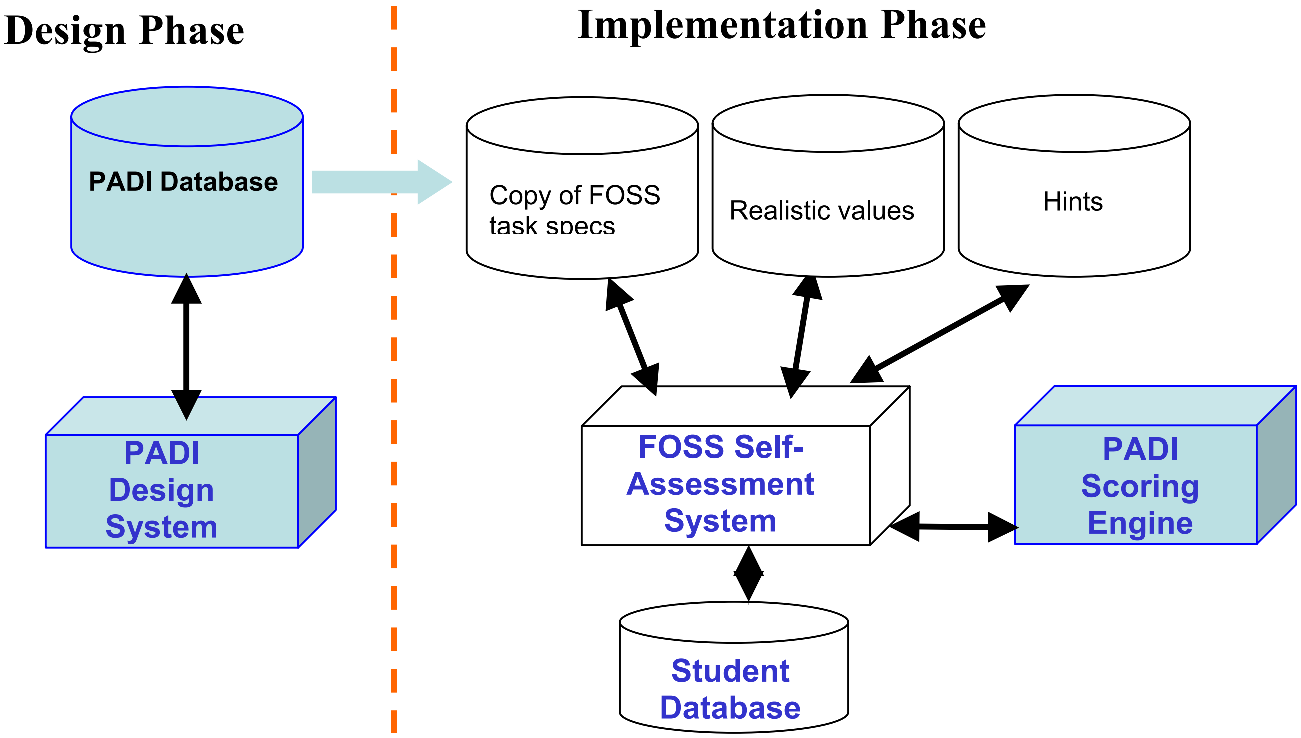


Figure 2. The overall architecture of the FOSS Self-Assessment System and the PADI Design System.

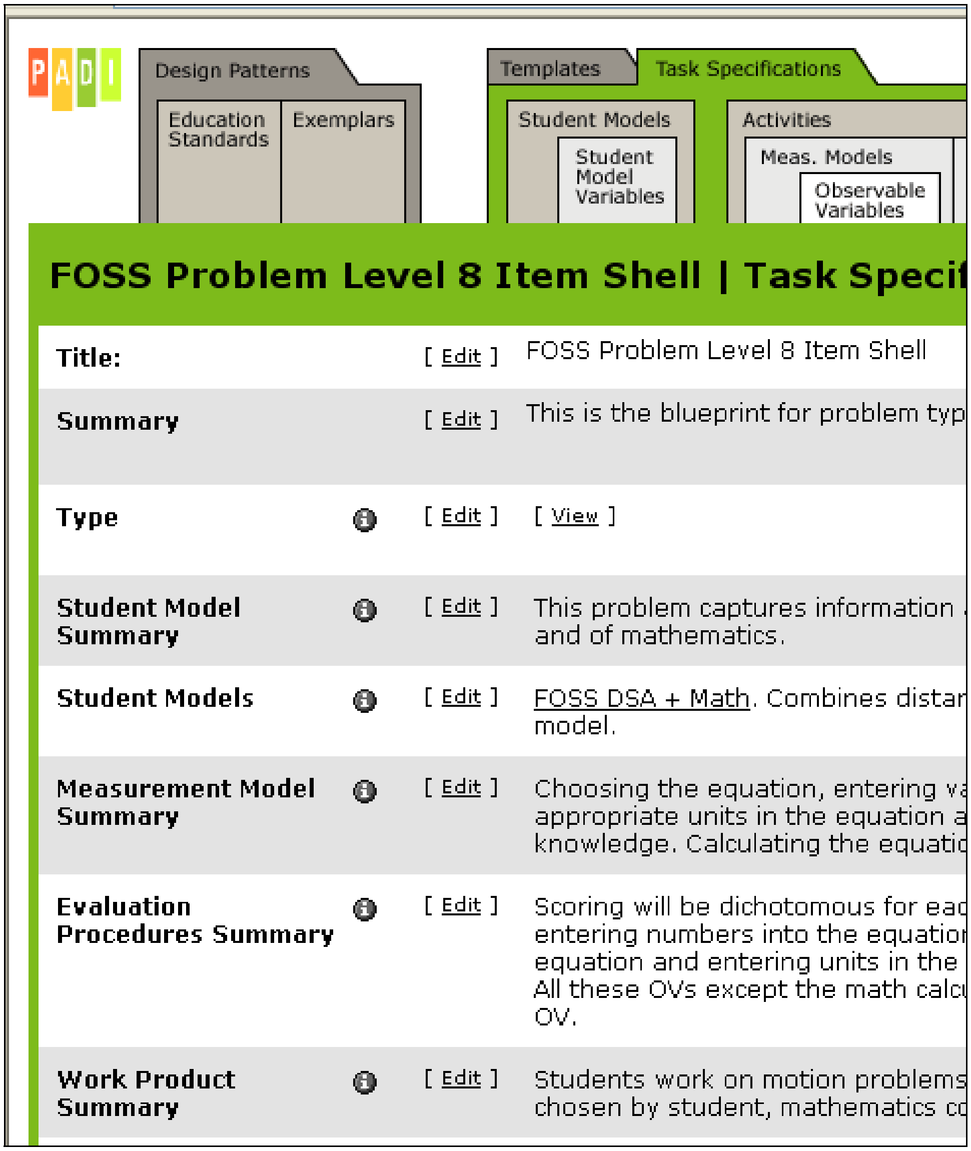
The next step in designing the task specifications was to layout the format of the problems students would encounter. Although specific problem levels had some variation in the number of questions that were asked in the problem, the general layout of the problems was the same. Each problem begins with a prompt and a graphical representation of the problem situation. Students are first asked to select the appropriate equation for solving the problem from a list of equations. Then, they fill in the equation with numbers and units from the problem text. Next, they solve the equation and use that solution to respond to the question posed in the problem prompt. Each response is considered evidence of student knowledge, and as such needs to be associated with the appropriate measure (DSA or Math). Designers determined that solving the equation required primarily mathematical ability, while the other responses required primarily DSA knowledge.

The layout of stimulus materials and the type of individual responses (i.e., selection from a list, constructed response, etc.) became parts of the task specification. The association of responses to measures also became part of the task specification so that proficiency estimates could be computed correctly from the response data when the Self-Assessment System is interacting with students. In PADI terms, a Measurement Model defines the way proficiency estimates are computed from the response data.

As mentioned above, the PADI Design System is used to design task specifications, or item shells, as they are called in FOSS. It is not an authoring or assessment delivery system. So, once all of the task specifications were finalized and entered into the PADI Design System, they were copied into the database of the FOSS Self-Assessment System. Then, when the FOSS Self-Assessment System needs to deliver an item to a student, it retrieves an item shell for the appropriate problem level from the local database, fills in the numeric parts with randomized values, and renders the item on the computer screen. It then gathers response data from student entries, and if, needed, packages the student response data and the task specification data (i.e., the measurement model part) to send to the PADI Scoring Engine. The PADI Scoring Engine takes that data and computes proficiency estimates using multidimensional item response modeling and returns proficiency estimates on each measure for the student.

Figure 3 is an excerpt from the task specification for a level eight problem. Note that the Student Model Summary indicates that items generated from this shell will produce evidence of students' knowledge of speed and of their mathematical ability. The Student Model, "FOSS DSA + Math," and a general summary of the Measurement Model is provided, along with an Evaluation Summary and a summary of Work Products. Details of the item layout are contained in an Activity object that is associated with the task specification.

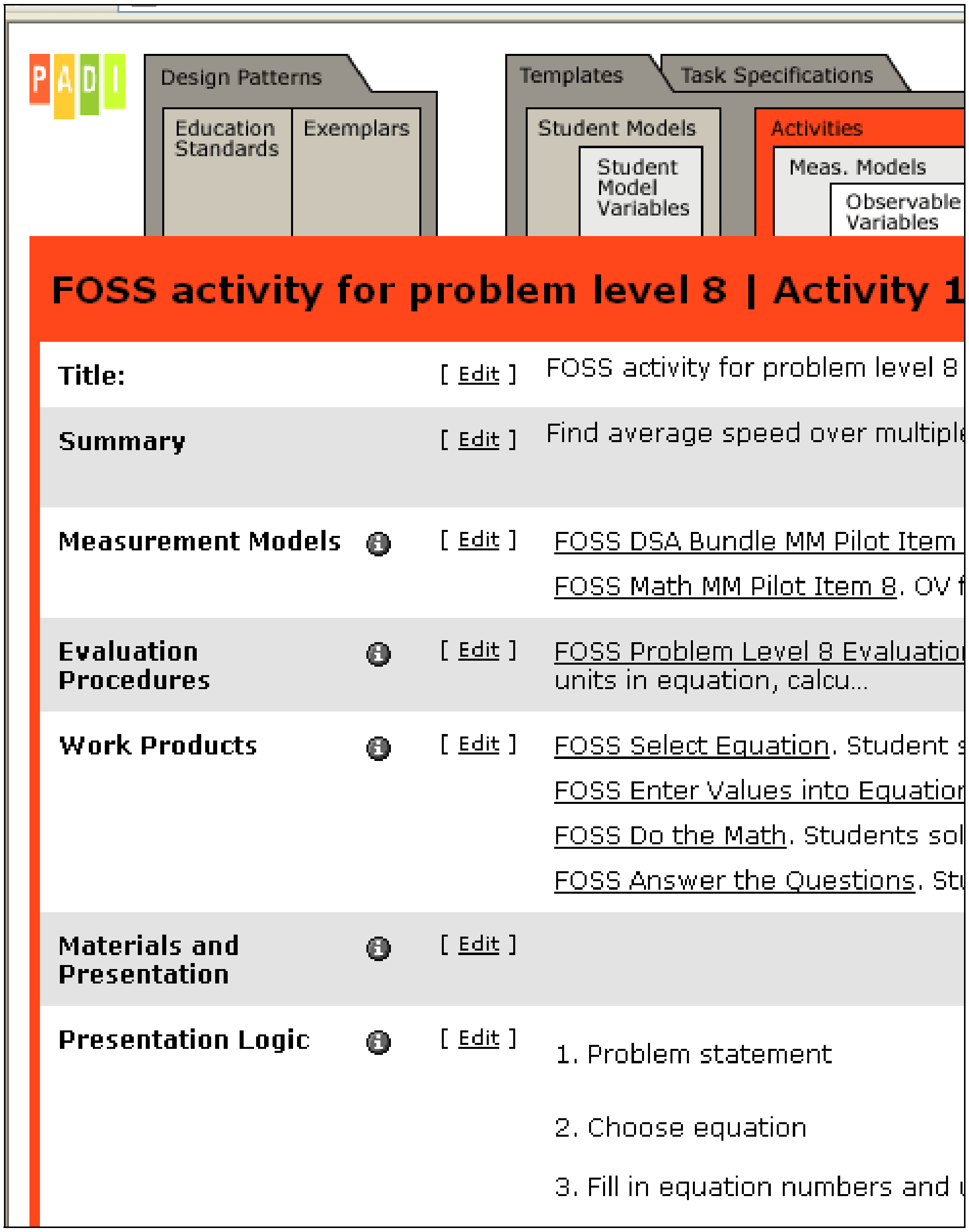
Every task specification contains at least one activity. In the FOSS Self-Assessment System, we generated exactly one activity for each task specification. An excerpt of the activity for the Problem Level Eight task specification is shown in Figure 4. Here, the details of the measurement models, which are used by the Scoring Engine, are specified, as are the evaluation procedures, the work products, and the way the item is to be presented to the student. All of this information is encoded in a format that is accessible by a computerized assessment authoring and delivery system, such as the FOSS Self-Assessment System.



FOSS Problem Level 8 Item Shell | Task Specif

Title:		[Edit]	FOSS Problem Level 8 Item Shell
Summary		[Edit]	This is the blueprint for problem typ
Type	ⓘ	[Edit]	[View]
Student Model Summary	ⓘ	[Edit]	This problem captures information and of mathematics.
Student Models	ⓘ	[Edit]	FOSS DSA + Math. Combines distar model.
Measurement Model Summary	ⓘ	[Edit]	Choosing the equation, entering va appropriate units in the equation a knowledge. Calculating the equatio
Evaluation Procedures Summary	ⓘ	[Edit]	Scoring will be dichotomous for eac entering numbers into the equation equation and entering units in the All these OVs except the math calcu OV.
Work Product Summary	ⓘ	[Edit]	Students work on motion problems chosen by student, mathematics co

Figure 3. Part of the task specification from the PADI Design System for a problem level eight item shell.

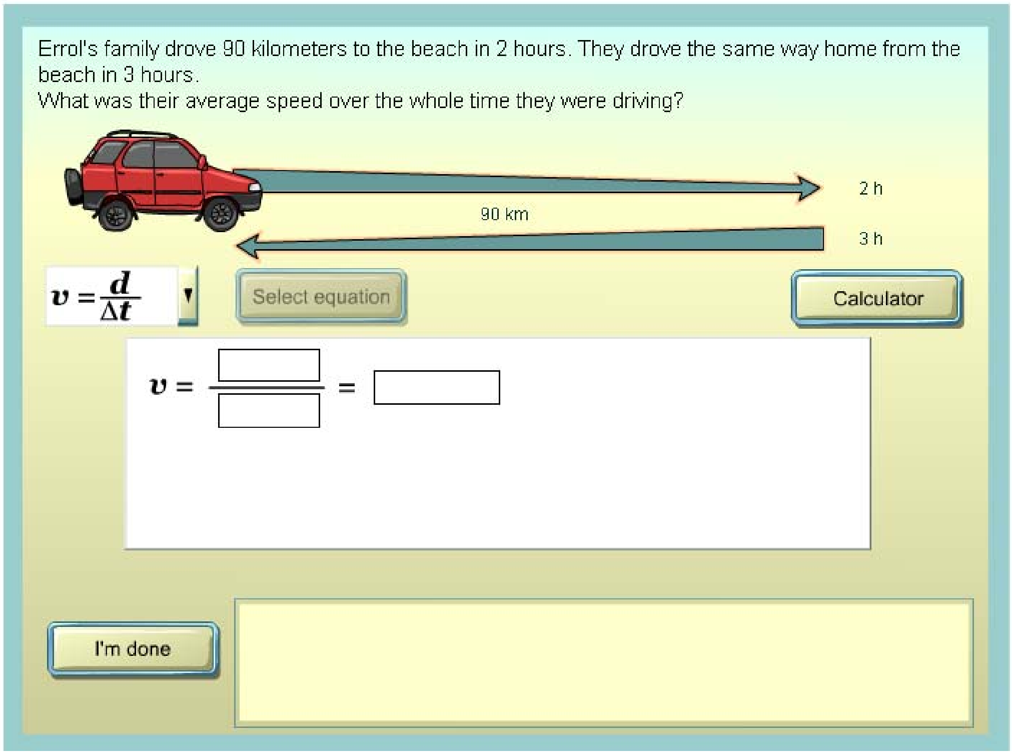


FOSS activity for problem level 8 | Activity 1

Title:	[Edit]	FOSS activity for problem level 8
Summary	[Edit]	Find average speed over multipl
Measurement Models	[Edit]	FOSS DSA Bundle MM Pilot Item FOSS Math MM Pilot Item 8. OV f
Evaluation Procedures	[Edit]	FOSS Problem Level 8 Evaluatio units in equation, calcu...
Work Products	[Edit]	FOSS Select Equation. Student s FOSS Enter Values into Equatio FOSS Do the Math. Students sol FOSS Answer the Questions. St
Materials and Presentation	[Edit]	
Presentation Logic	[Edit]	1. Problem statement 2. Choose equation 3. Fill in equation numbers and

Figure 4. Part of the activity from the problem level eight task specification contained in the PADI Design System.

An example of an item at problem level eight as generated and rendered to a student is shown in Figure 5.



Errol's family drove 90 kilometers to the beach in 2 hours. They drove the same way home from the beach in 3 hours.
What was their average speed over the whole time they were driving?

Figure 5. Screen shot of a problem at level eight.

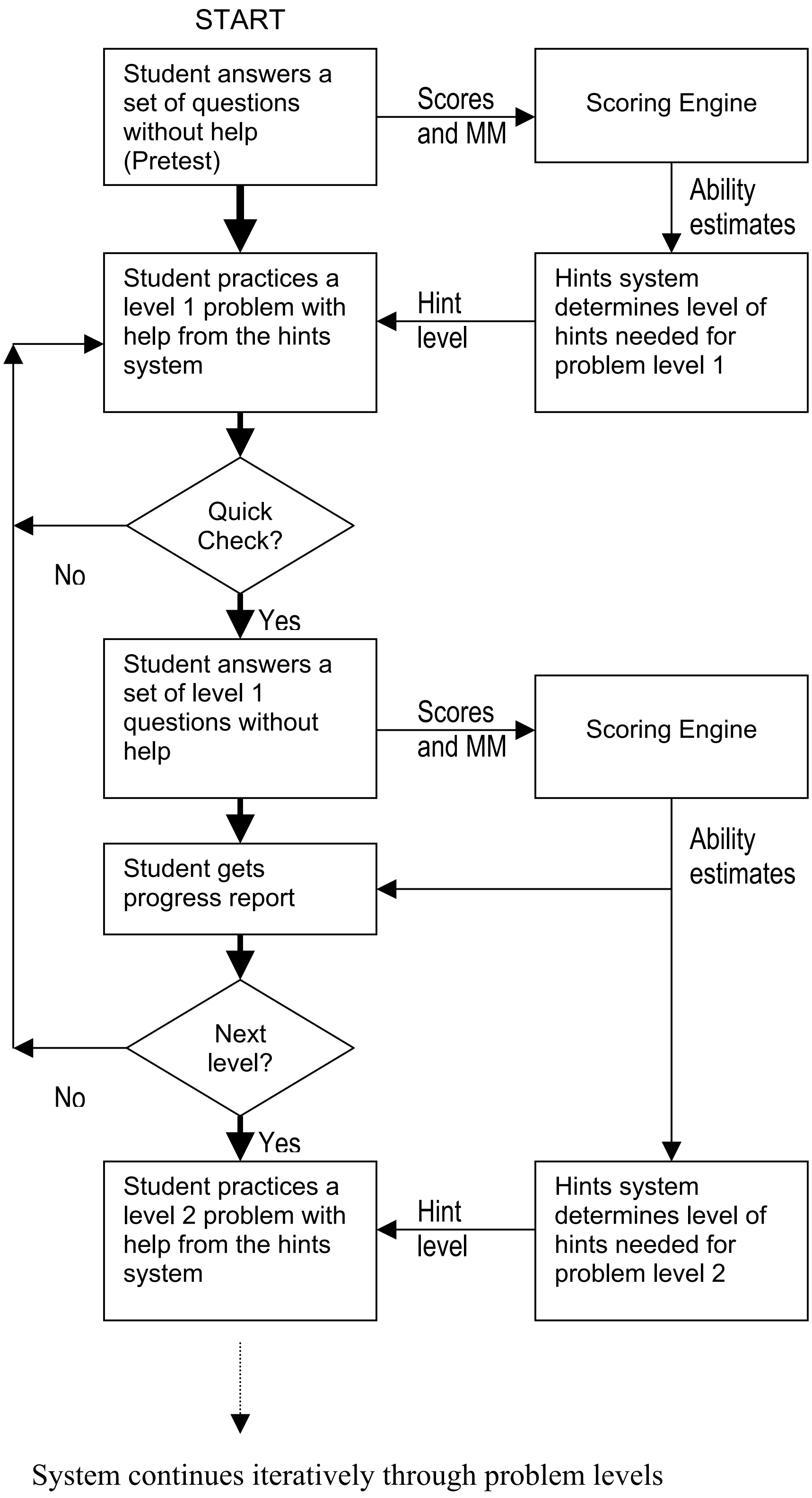
As can be seen in Figure 5, the shell for an item has several components. At the top is the statement of the problem. The text of the problem is constant across problems, but the values of the given information, such as the distance and the time traveled in problem eight, are randomly generated from a list of realistic values for each item. The values are stored as pairs to ensure that appropriate values result each time. The other component is the drop-down selection list of the possible equations to use to solve the problem. In this prototype, there were three options in the list; *distance = velocity x time*, *velocity = distance/time*, and *time = distance/velocity*. After selecting an equation from the drop-down menu, the student presses the "select equation" button, which then causes the equation to be represented in another part of the item shell, the white "work space" area. The equation representation in the workspace has input boxes for each of the equation values and for the calculated answer values. The student determines from the problem statement and graphic the values to be entered and types them into the workspace area along with the units. The student then makes the calculation pertinent to the equation, and enters that value and the units into the box for the result. Upon completion, the student clicks on the "I'm done" button to end the problem.

Students are scored on two variables in each problem; one that measured their understanding of speed (as a component of a larger distance, speed and acceleration knowledge variable) and one that measured their mathematical ability. Scores were assigned for each of the relevant steps in the problem. For example, in problem type eight shown in Figure 5, students are scored on the following problem steps:

	Variable for scoring	
Problem step	Speed problem-solving	Mathematics
Selects correct equation	**X**	
Places all the correct values into the equation	**X**	
All values in the equation have the correct units	**X**	
Calculated value is correct		**X**
Calculated value has correct units	**X**	

The automated help system

Figure 6 shows the architecture of the Self-Assessment System that includes the automated help system. The first time that a student logs in to the system, he or she takes the pretest. Throughout the pretest, the system records the student's responses to each question, tracking separately the responses that map to the content knowledge variable and to the mathematics variable. At the conclusion of the pretest, the system sends the scores on both variables to the PADI Scoring Engine, which resides on an external server and is part of the PADI system. The statistical model of the evidence rules that relate how student scores on the tasks (problems) should be interpreted are contained in the measurement model components of the task specifications stored in the Self-Assessment System database.


START
Student answers a set of questions without help (Pretest)
Scores and MM
Scoring Engine
Ability estimates
Student practices a level 1 problem with help from the hints system
Hint level
Hints system determines level of hints needed for problem level 1
Quick Check?
No
Yes
Student answers a set of level 1 questions without help
Scores and MM
Scoring Engine
Ability estimates
Student gets progress report
Next level?
No
Yes
Student practices a level 2 problem with help from the hints system
Hint level
Hints system determines level of hints needed for problem level 2

System continues iteratively through problem levels

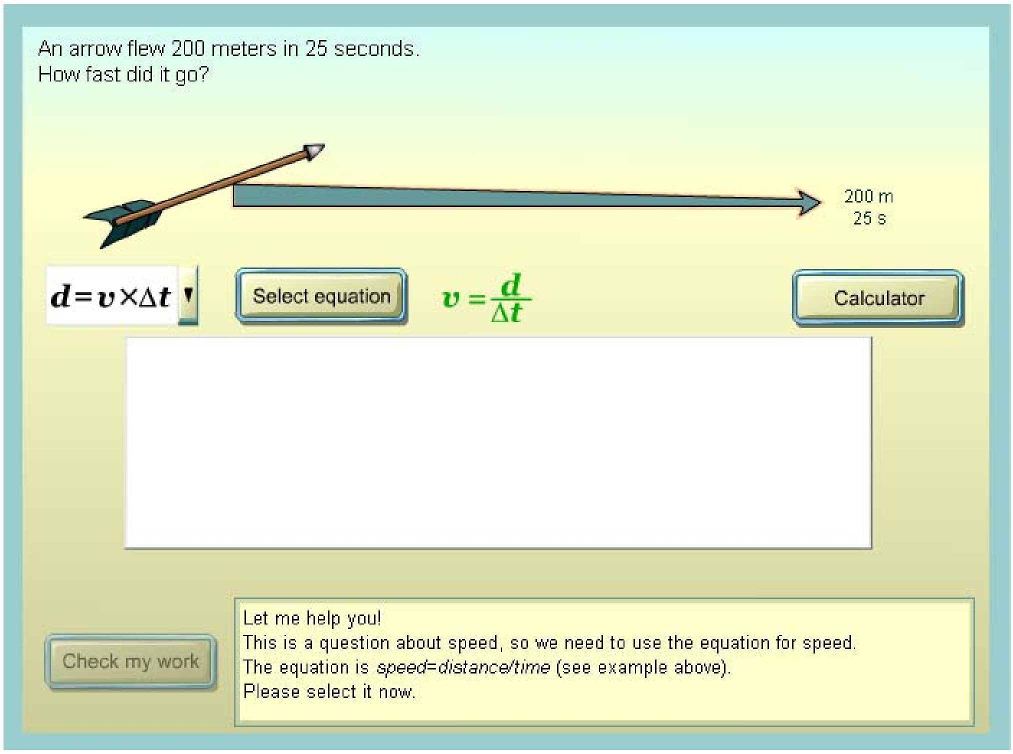
Figure 6: Flowchart of the FOSS Self-Assessment System logic.

Using this statistical model, the Scoring Engine calculates a proficiency estimate for each student model variable, which represents the student's current ability level. These values are returned to the Self-Assessment System. The Self-Assessment System maintains a history of these estimates each time they are generated for a student. The student's most recent ability estimate (e.g., from the pretest or from a prior Quick Check) is used to determine the type of hints that the student will receive in the next round of practice problems.

After the pretest, the student begins the practice problems at level one. During the practice session hints are available to the student. Generally, the student has control over when to have their work checked, as described shortly. One exception to this is at the start of the problem. If the student has not selected the correct equation to solve the problem and tries to add it to the workspace by clicking the "Select Equation" button, the system offers a hint. An example of how a hint is provided to a student who has selected the wrong equation is shown in Figure 7. If the student did not choose the correct equation, a hint appears in the field at the bottom right hand section of the screen, next to the "check my work" button. Hints are text-based and are designed to point out to the learner the features of the problem he or she should pay attention to. The system gives more or less detailed help depending on the hint stream that has been selected based on their latest ability estimate. In addition to the written hint in the text box there are other visual prompts with some hints.

Hints are selected based on the ability estimate of the student. For each problem there are three "streams" of hints. The level 1 hint stream is for higher ability learners who need the least amount of help, level 2 is for medium ability learners who need intermediate help, and level 3 is for learners of lower ability who need deeper, more concrete hints. In Figure 7, the hint shown is from the level 3 hint stream, the most helpful level of hints. The written hint tells the student that he or she needs to select a different equation. Because it is a level 3 hint, the text specifies the equation to select and, for additional reinforcement, a visual hint in the form of the equation being shown in green alongside the "select equation" button. This initial hint repeats until the student selects the correct equation. The rationale for preventing students from proceeding through the problem after selecting the wrong equation is that it is a waste of instructional time to allow them to continue. A student cannot be guided to a successful solution of the problem if he or she is applying the wrong equation, so feedback on any other steps beyond that could imply that the student was on a correct solution path and thus confuse them.

Other than that initial, unsolicited check of their progress, hints are under the control of the student. Whenever the student clicks on the "Check My Work" button, the system checks her work in a systematic way. In other words, the learner has control of when to seek feedback. The student may, if she wants, proceed to the end of the problem before clicking the "Check My Work" button.



An arrow flew 200 meters in 25 seconds.
How fast did it go?

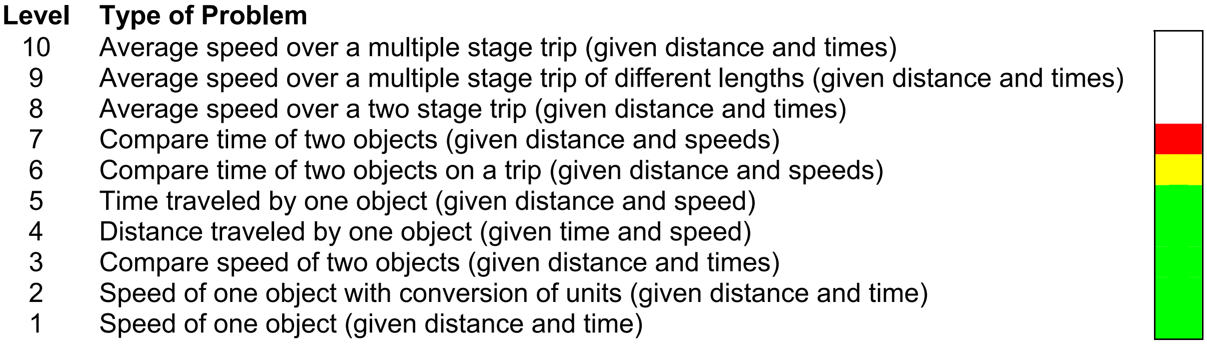
Let me help you!
This is a question about speed, so we need to use the equation for speed.
The equation is *speed=distance/time* (see example above).
Please select it now.

Figure 7. Screen shot of a "Practice" problem showing the feedback the student received.

Once invoked, the help system checks the work according to each of the scorable steps of the problem that are listed on page 18. The student continues through the problem, having the system check their work as frequently as she wants.

After completing at least one problem as a practice activity at the current level of problems, the student may opt to take a "Quick-Check." The student decides if he or she wants to practice on more than one problem. Students can practice as much as they feel is necessary before they opt to take the quick check. The level of hints stays the same until the student takes a Quick Check because the student's ability estimate stays the same (i.e., it is not recomputed). After taking a Quick Check, the student's ability estimate is updated and the hint level may change.

When a student selects the "Quick Check" button, the system delivers three problems at the student's current level of performance. So, if the student is at the Problem 1 level, she will get three problems that use exactly the same item shell that she has just been practicing on, although the values are still randomly generated, and the system ensures that she does not get the exact same problem. In these Quick Check problems, the student gets no hints, even though she may make mistakes. When the student has completed each quick check problem she clicks on an "I'm done" button to continue to the next problem in the set. At the completion of the Quick Check items, the Self-Assessment System, which has been capturing the scores at each problem step, compiles the scores and sends them to the scoring engine. The scoring engine calculates the student's ability based upon the latest scores from the Quick-Check and returns the estimate to the self-assessment program. This estimate is used to determine the level of hints that the student will be given for the next level of practice problems.



Level	Type of Problem	
10	Average speed over a multiple stage trip (given distance and times)	⬜
9	Average speed over a multiple stage trip of different lengths (given distance and times)	⬜
8	Average speed over a two stage trip (given distance and times)	⬜
7	Compare time of two objects (given distance and speeds)	🟥
6	Compare time of two objects on a trip (given distance and speeds)	🟨
5	Time traveled by one object (given distance and speed)	🟩
4	Distance traveled by one object (given time and speed)	🟩
3	Compare speed of two objects (given distance and times)	🟩
2	Speed of one object with conversion of units (given distance and time)	🟩
1	Speed of one object (given distance and time)	🟩

Figure 8. A student progress report.

The student also gets feedback after each quick check in the form of a report as shown in Figure 8. The report is based upon the student ability estimate on each variable and maps the student progress against certain learning benchmarks on the dimensions being measured. Progress is indicated to the student in a simple traffic light analogy. A green “light” indicates that the student has done well enough at that level for her to proceed. A yellow light means more practice at that level is advised. A red light tells the student that she is strongly recommended not to proceed beyond that level without more practice. Although a student is advised to practice more when she gets a yellow or red light on her report, the system allows the student to proceed if she wishes. However, the teacher of the class is also able to review the student’s report and so could. Students progress iteratively through ten levels of problems of increasing difficulty in the Self-Assessment System.

Conclusion

The PADI design system provides a method for designing and storing detailed task specifications that can then be used by an assessment authoring and delivery application to generate tasks, render them to students, gather responses and provide feedback. The PADI system includes a Scoring Engine to compute student proficiencies from response data and task specifications. The FOSS Self-Assessment System uses the task specifications as item shells to produce unique problems to students. It also accesses the Scoring Engine to produce ability estimates that are used to make decisions about the level of hints that students are most likely to need. The system is currently undergoing a field test to determine the efficacy of this approach to delivering hints that are tailored to individual learners and their needs at a particular time in their use of the interactive learning environment.

References

Aleven, V., & Koedinger, K. R. (2000). *Limitations of Student Control: Do Students Know When They Need Help?* Paper presented at the Intelligent Tutoring Systems: 5th International Conference, Montreal, Canada.

Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. (2003). Help Seeking and Help Design in Interactive Learning Environments. *Review of Educational Research, 73*(Fall 2003), 277-320.

Arroyo, I., Beck, J. E., Beal, C. R., Wing, R. E., & Woolf, B. (2001). *Analyzing students' response to help provision in an elementary mathematics Intelligent Tutoring System.* Paper presented at the Workshop on Help Provision and Help Seeking at the Tenth International Conference on Artificial Intelligence in Education. San Antonio, TX.

Arroyo, I., Beck, J. E., Woolf, B., Beal, C. R., & Schultz, K. (2000). *Macroadapting Animalwatch to gender and cognitive differences with respect to hint interactivity and symbolism.* Paper presented at the Fifth International Conference on Intelligent Tutoring Systems, Montreal, Canada.

Dutke, S., & Reimer, T. (2000). Evaluation of two types of online help information for application software: Operative and function-oriented help. *Journal of Computer Assisted Learning, 16*, 307-315.

Hofer, M., Niegemann, H. M., Eckert, A., & Rinn, U. (1996). Pedagogische Hilfen fur interaktive selbstgesteurte Lernprozesse und Konstruktion eines neuen Verfahrens zur Wissensdiagnose [Instructional help for interactive self-directed learning processes and construction of a new procedure for knowledge diagnosis]. *Zeitschrift fur Berufs- und Wirtschaftspadagogik Beiheft, 13*, 53-67.

Johnson, W. L., Shaw, E., Marshall, A., & LaBore, C. (2003). *Evolution of User Interaction: The Case of Agent Adele.* Paper presented at the IUI'03, Miami, FL.

Katz, S., Allbritton, D., & Connelly, J. (2002). Going Beyond the Problem Given: How Human Tutors Use Post-Solution Discussions to Support Transfer. *International Journal of Artificial Intelligence in Education, 13*.

Mandl, H., Graesel, C., & Fischer, F. (2000). Problem-oriented learning: Facilitating the use of domain-specific and control strategies through modeling by an expert. In W. J. Perrig & A. Gob (Eds.), *Control of human behavior, mental processes and consciousness* (pp. 165-182). Mahwah, NJ: Erlbaum.

McKendree, J. (1990). *Effective Feedback Content for Tutoring Complex Skills.* Paper presented at the HUMAN-COMPUTER INTERACTION.

Roberts, B., Pioch, N., & Ferguson, W. (2000). Verbal Coaching During a Real-time Task. *International Journal of Artificial Intelligence in Education, 11*, 377-388.

Wood, D., & Middleton, D. (1975). A study of assisted problem solving. *British Journal of Psychology, 66*, 181-191.