

# All you need to know about ITS (in 60 slides)

Created by Kurt VanLehn ©



Presented by Vincent Alevan



VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227-265.

## Terminology (slide 1 of 5)

- ◆ **Task domain:** The information and skills being taught by the tutor
- ◆ **Task:** A multi-minute activity that can be skipped or interchanged with other tasks.
- ◆ **Step:** Completing a task takes many steps. Each is a user interface event.



The sun exerts a gravitational force on the earth as the earth moves in its orbit around the sun. Does the earth pull equally on the sun? Explain why.

The task

Each tutor turn + student turn in the dialogue is a step

Student input is the 2<sup>nd</sup> half of the step

Log of previous turns:

moves in its orbit around the sun. Does the earth pull equally on the sun? Explain why.  
 Student:  
 Tutor: Is anything you can add to this?  
 Student:  
 Tutor: Kind of  
 Tutor:  
 Tutor: How does a law of motion apply to this situation?  
 Tutor:

## Example from Steve

- ◆ **A knowledge component:** If you are starting the air compressor, first check all the indicator lights by pressing their “lamp test” buttons.
- ◆ **A learning event:**
  - Steve: Go on.
  - Student: <Opens sea water valve>
  - Steve: You forgot some steps. Hint: check your lamps.
  - Student: <pushes “lamp test” button under “Low oil” indicator>
  - Steve: Good. Please continue.
- ◆ **Correct** = consistent with US Navy doctrine

## Terminology (slide 2 of 5)

- ◆ **Knowledge component:** A task domain concept, principle, fact, etc. Any fragment of the persistent, domain-specific information that should be used to accomplish tasks.
- ◆ **Learning event:** A mental event; the construction or application of a knowledge component, often while trying to achieve a task.
- ◆ **Incorrect:** Inconsistent with the instructional objectives of the tutor.

## Terminology (slide 3 of 5)

Until done tutoring, do:

- Tutor poses a **task**
- Until task is achieved, do:
  - » Tutor may give a hint
  - » Student does a **step**
  - » Tutor may give feedback
- Tutor & student may discuss the solution
  - » Usually, step by step

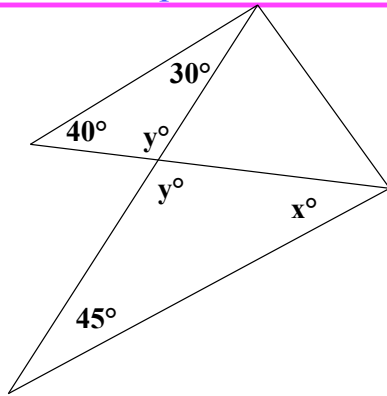
◆ **Outer loop** over tasks

Outer loop = Problem selection

◆ **Inner loop** over steps

Inner loop = Within-problem step-by-step guidance

## Tutoring systems with only the outer loop: Called computer aided instruction (CAI), CBT...

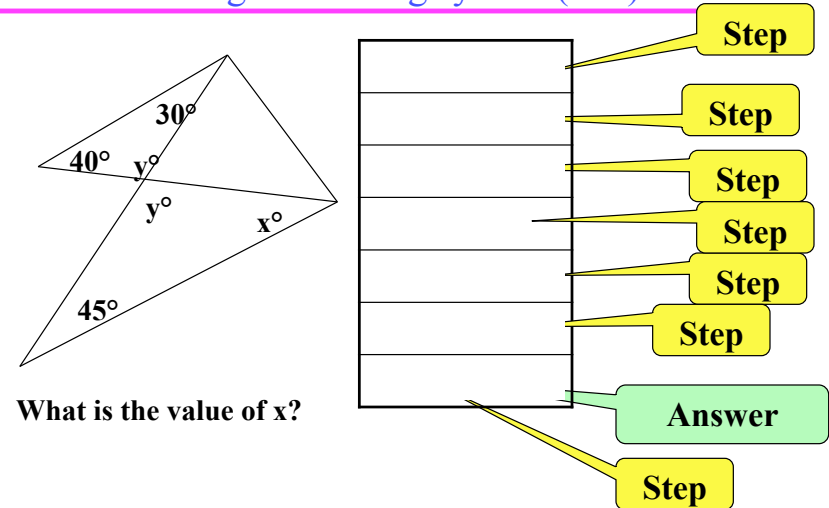


$x =$

**Answer**

What is the value of  $x$ ?

## Tutoring systems with both inner & outer loops: Called intelligent tutoring system (ITS)



“Systems that lack an inner loop are generally called Computer-Aided Instruction (CAI), Computer-Based Training (CBT) or Web-Based Homework (WBH). Systems that do have an inner loop are called Intelligent Tutoring Systems (ITS).”

VanLehn, 2006 (p. 233)

## Outline: Design issues for ITS

- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks to select from
- ◆ Inner loop
- ◆ Implementation and scale-up

## Some task selection techniques

(next 6 slides)

---

1. Student selects task from a menu
2. Prescribed task sequence
3. Mastery learning
4. Macro-adaptation

## 2. Follow a prescribed task sequence

---

- ◆ Example
  - Study chapter
  - Study 3 examples
  - Solve 6 problems with help from system
  - Solve 3 problems without help
  - Study next chapter
  - Etc.
- ◆ Student must complete the work on one chapter before moving to the next

## 1. Student selects task from menu

---

- ◆ Easy way of giving instructor control over task assignment (though not the only way, see *the Tutorshop*)
  - Different instructors assign different tasks
  - Can assign different students different tasks
  - Speed up / slow down
- ◆ Students can explore, do extra work, etc.

## 3. Mastery Learning

---

- ◆ For each chapter C do:
  - Study text and examples of chapter C
  - Do each required task of C
  - Until mastery is reached, do another task of C
- ◆ Different students do different numbers of tasks, but all achieve mastery eventually
- ◆ Requires assessing student's competence
  - Inner loop does the assessment, or
  - Tests

## 4. Macro-adaptation

---

- ◆ Same control structure as Mastery Learning
  - But fewer tasks required to reach mastery
- ◆ Basic idea: choose a task that requires just a few unmastered knowledge components
  - Tasks are described by the knowledge components that they address
  - System assesses the student's mastery of individual knowledge components
  - System chooses a task that matches the student's needs

In the Cognitive Tutor literature, the term "Cognitive Mastery Learning" is used for the notion that VanLehn calls "Macro-adaptation."

## Student model

---

- Student model = persistent data on the student
- Lousy name, but traditional
  - ◆ Which tasks have been done
  - ◆ Students' knowledge at this moment (i.e., competence so far)
  - ◆ Learning styles, preferences, etc.
    - E.g., visual vs. verbal
  - ◆ Demographics (e.g., SAT, College Major)

## How to represent student's knowledge? Depends on the task selection method!

---

- ◆ Macro-adaptive task selection
  - Need mastery of each knowledge component
- ◆ Mastery learning
  - Which chapter is being studied now
  - Competence (a single number) on this chapter
- ◆ Choose next task on the list
  - Student model has just a pointer into the list
- ◆ Student chooses the next task
  - No student model of knowledge needed

## Obtaining a large set of tasks

---

- ◆ Task generation
  - E.g., fill in a template e.g.,  $A*x + B = C$
  - E.g., install a fault in a piece of equipment
  - Necessary in some training applications
  - Hard to control difficulty, coverage, etc.
- ◆ Task authoring
  - Author fills out a form
  - Authoring system does as much work as possible
    - » Solves the task
    - » Evaluates difficulty
    - » Evaluates coverage of the task domain

## Outline: Design issues in ITS

---

- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop ← Next
  - 5 common services
- ◆ Implementation and scale-up

## Inner loops often offer services

---

- ◆ Designed to help students learn: Scaffolding
- ◆ Common services (covered in this talk)
  - Minimal feedback (correct/incorrect) on a step
  - Error-specific feedback on an incorrect step
  - Hints on the next step
  - Assessment of student's knowledge.
  - Review of the student's solution
- ◆ Tutor-specific services
  - E.g., physically impossible views of an underwater robot
- ◆ Warning: Any service can be misused!
  - Often called “gaming the system”

## Minimal feedback: How?

---

- ◆ Andes: Color the step red or green
- ◆ Steve: Shake head “yes” or “no”
- ◆ AutoTutor: Start tutor's turn with
  - “No...”
  - “Well...”
  - “Okay...”
  - “Yeah...” or
  - “Great!”

## Minimal feedback: What counts as incorrect?

---

- ◆ Andes: Should it have Yellow steps for correct but useless equations?
- ◆ Sherlock: Better to waste time or waste parts?
  - “Instead of replacing circuit board G1X2, some experts would have tested it. Here's how.”
- ◆ Pyrenees: Skipping steps is incorrect for novices but allowed for intermediates.
- ◆ CTAT: Should out-of-order steps be marked wrong?

## What if the step cannot be classified as correct or incorrect?

---

- ◆ Cognitive tutors: If students step can't be recognized, it's *incorrect*
- ◆ SQL tutor: If student's step can't be recognized, it's *correct*

## Minimal feedback: When?

---

- ◆ Fixed policies
  - Immediate – as soon as step is entered
  - Demand – wait until student asks
  - Delayed – wait until problem is submitted
    - » Common with real-time skills, e.g., air combat
- ◆ Adaptive
  - Fading – changes policy as competence increases
  - Decision theoretic – Decide at each error whether feedback maximizes expected utility

## Outline: Design issues in ITS

---

- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop
  - Minimal feedback on a step
  - **Hints on the next step**
  - Error-specific feedback on an incorrect step
  - Assessment of student's knowledge.
  - Review of the student's solution
- ◆ Implementation and scale-up



Next

## Next step hints: When? Some fixed policies are...

---

- ◆ Andes:
  - Give hint only when the student asks
- ◆ Cognitive Tutors:
  - When student has failed several times and has not yet asked for a hint, give them one
- ◆ Steve:
  - Before even attempting the step, if the step is new or student failed miserably last time, then give them a hint.

## Next step hints: Which step to hint?

- ◆ A correct step, and
- ◆ A step not yet done by student, and
- ◆ A step that the instructor would prefer, and maybe
- ◆ A step that is consistent with the student's plan
  - Requires plan recognition (can be hard AI problem)
  - Pedagogical benefit unknown
    - » Would make a good *in vivo* experiment

## Which step to hint? CTAT's method for example-tracing tutors:

- ◆ Select leading interpretation of student behavior (i.e., the interpretation closest to the preferred path through the problem)
- ◆ If the student just made an error, and it was not an out-of-order error with respect to the leading interpretation, and there are hints related to this step then give a hint on the step on which the error occurred
- ◆ Else if the student selected an interface element, and within the leading interpretation a step in this interface element is a valid next step, and there are hints for this step, then give a hint on this step
- ◆ Else, give a hint on the first unvisited step closest to the start state that is not out-of-order within the leading interpretation, and has hints

## Next step hints: How?

- ◆ Hint sequences
  - Forward button prints next hint
  - Back button
  - Done button lets student go back to entering steps
- ◆ Start in middle of sequence (contingent tutoring)
- ◆ Dialogues (usually menu-based) e.g.,
  - Andes: What quantity is the problem seeking?
  - Student: Acceleration of the car at T2
  - Andes: Good. What physics principle will you use?
  - Student: Newton's second law.
  - Andes: Good. You've already done the first step of applying Newton's law. The second step is...

## Outline: Design issues in ITS

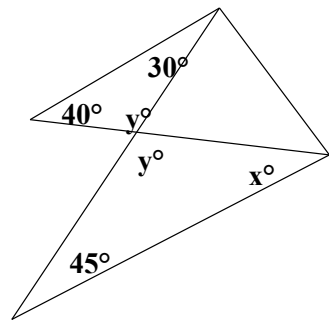
- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop
  - Minimal feedback on a step
  - Hints on the next step
  - **Error-specific feedback on an incorrect step**
  - Assessment of student's knowledge.
  - Review of the student's solution
- ◆ Implementation and scale-up



Next

## Error specific feedback: How?

### Usually the first few hints of a hint sequence



What is the value of  $x$ ?

$$40+30+y=180$$

$$y = 250$$

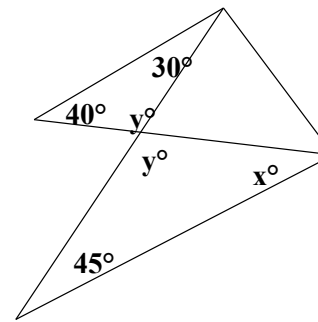
**Oops! Check your arithmetic.**



OK



## Error-specific feedback becomes more specific



What is the value of  $x$ ?

$$40+30+y=180$$

$$y = 250$$

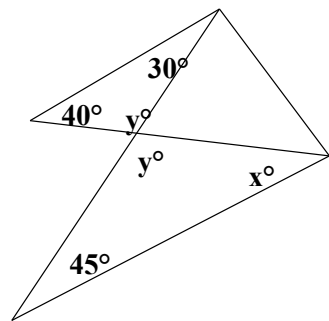
**You seem to have made a sign error.**



OK



## Hints segue from error specific feedback to next-step hinting



What is the value of  $x$ ?

$$40+30+y=180$$

$$y = 250$$

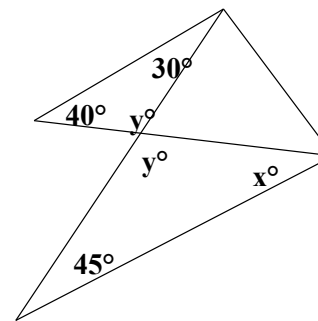
**Try taking a smaller step.**



OK



## Next-step hints become more specific



What is the value of  $x$ ?

$$40+30+y=180$$

$$y = 250$$

**Try doing just one arithmetic operation per step.**



OK



Def: A *bottom-out hint* is the last hint, which tells the student what to enter.

What is the value of  $x$ ?

$$40+30+y=180$$

$$y = 250$$

**Enter  $70+y=180$ , and keep going from there.**

◀ OK ▶

## Error specific feedback: How to recognize errors?

- ◆ Many error recognizers must be constructed by...
  - Hand-authored from student errors
  - Induced from student errors via machine learning
  - Theory-based error generators
- ◆ Only worth doing if error-specific feedback is more effective than minimal feedback & next-step hints alone
  - Are errors corrected during the error-specific hints in a hint sequence?
  - Do errors corrected then re-occur less? Self-corrected more?
  - Another good in vivo experiment

## Outline: Design issues in ITS

- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop
  - Minimal feedback on a step
  - Hints on the next step
  - Error-specific feedback on an incorrect step
  - **Assessment of student's knowledge.**
  - Review of the student's solution
- ◆ Implementation and scale-up

Next

## Terminology (slide 4 of 5)

- ◆ An *assessment* judges a student
  - How competent?
  - What knowledge?
- ◆ An *evaluation* judges a tutoring system
  - Formative – what needs to be fixed?
  - Summative – how effective compared to control?
  - Parametric – why is it effective?

## ITS assessments: Who needs them?

- ◆ Tutoring system (maybe)
  - Used some (not all) outer loop task selectors
  - Used by adaptive feedback & hinting
- ◆ Students (definitely)
- ◆ Teachers (but rarely for grading)
  - Can't defend what they didn't do themselves.
- ◆ “The testing establishment” (not yet)

## Assessments: Granularity

- ◆ Large grained assessments
  - Single number
  - Based on evidence from a whole course
  - Good for making big decision (e.g., pass/fail)
- ◆ Fine grained assessments
  - One number per knowledge component
  - Based on learning events for only that component
  - Good for making small decisions
    - » Next task
    - » Next hint

SELECT		covered: 33%, learned: 32%
FROM		covered: 41%, learned: 39%
WHERE		covered: 10%, learned: 9%
GROUP BY		covered: 58%, learned: 58%
HAVING		covered: 2%, learned: 2%
ORDER BY		covered: 44%, learned: 44%

## Assessments: How?

- ◆ See Ryan Baker's overview of Knowledge Tracing
- ◆ Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253-278.

## Outline: Design issues in ITS

- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop
  - Minimal feedback on a step
  - Hints on the next step
  - Error-specific feedback on an incorrect step
  - Assessment of student's knowledge.
  - **Review of the student's solution**
- ◆ Implementation and scale-up

Next

## Why review solutions?

- ◆ Don't want to interrupt the task
  - Real-time skills
  - Immediate feedback and hints have been faded out
  - So students must detect and correct own errors
  - Little headroom for conceptual learning during problem solving?
- ◆ Hindsight is often clearer
  - Can see effects of bad strategic choices
- ◆ Students more likely to focus on learning, instead of focusing on finishing the task

## SQL tutor exemplifies mixed control

Student selects level of review:

1. The number of errors.
2. Minimal feedback (color) on the tutor's favorite error
3. Error-specific feedback on the tutor's favorite error.
4. More detailed error-specific feedback on the tutor's favorite error.
5. All errors: A list containing a description of every error.
6. Partial solution: The correct version of the clause where the tutor's favorite error appeared.
7. Complete solution: The ideal solution to the problem.

## The review can be a long discussion: Who controls it?

- ◆ Tutor led
  - Katz's extension to Andes: Tutor leads a dialogue
- ◆ Student led
  - Display a list of the student's steps
  - Display minimal feedback (color) on each one
  - Student can click on any step for a discussion of it
    - » Typically clicks on an incorrect step

## Outline: Design issues in ITS

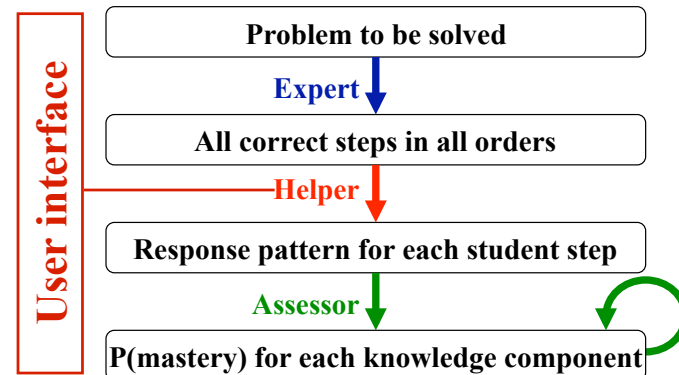
- ◆ Outer loop
  - How to select the next task
  - How to obtain a large set of tasks
- ◆ Inner loop
  - Minimal feedback on a step
  - Hints on the next step
  - Error-specific feedback on an incorrect step
  - Assessment of student's knowledge.
  - Review of the student's solution
- ◆ Implementation and scale-up

Next

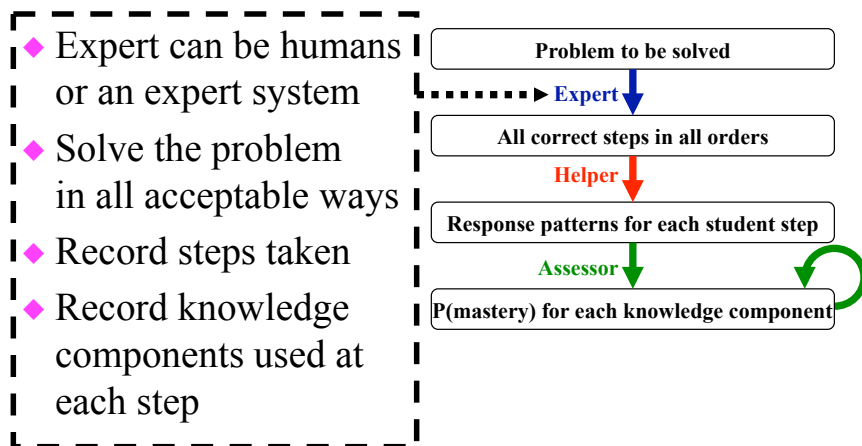
## Terminology (slide 5 of 5)

- ◆ The *response pattern* of a student's step represents how a step was accomplished by the student
- ◆ E.g., 4 response patterns:  
(S = student; T = tutoring system)
  1. S: Correct.
  2. S: Error; T: Hint; S: Correct.
  3. S: Help; T: Hint; S: Correct.
  4. S: Help; T: Hint; S: Help; T: Hint; S: Help; T: Bottom out hint; S: Correct

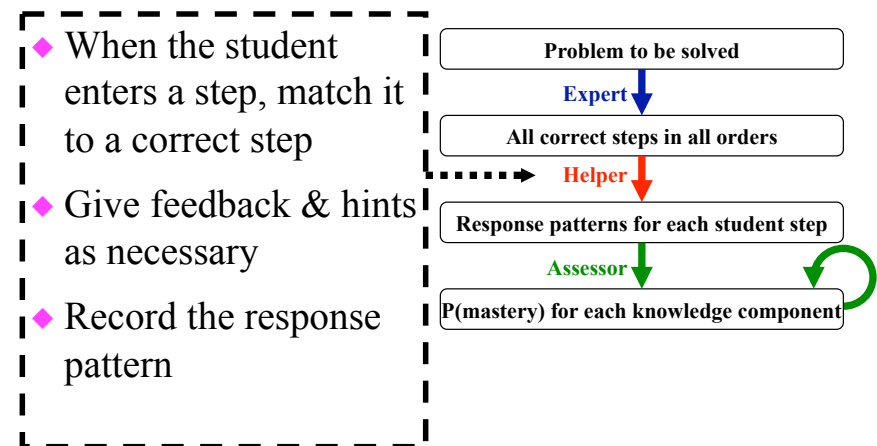
## The major components of an ITS (not really: conceptual only)



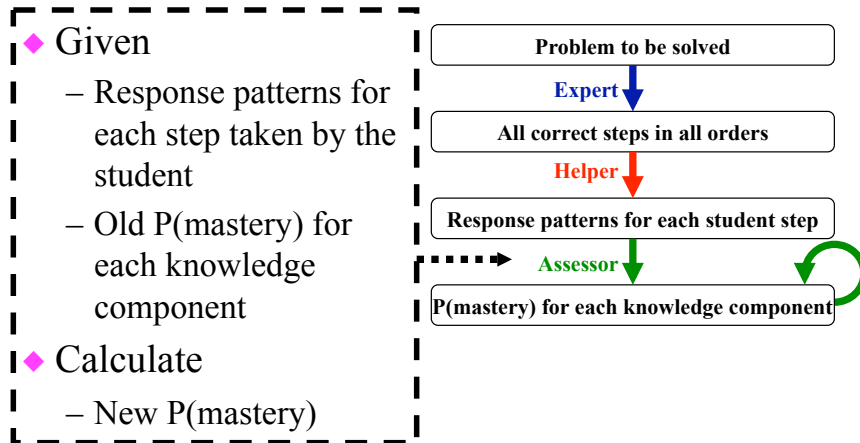
## The expert's computation



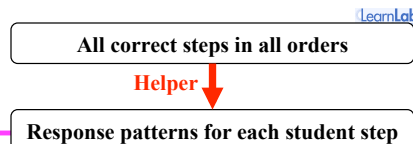
## The helper's computation



## The assessor's computation

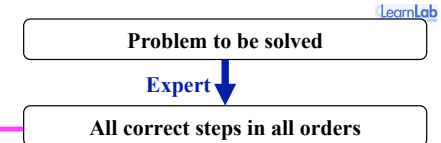


## 3 main design issues for the Helper



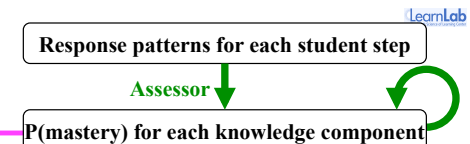
- ◆ Matching student's step to correct steps
  - Menus: trivial
  - Math expressions: Substitute numbers for variables
  - Physical actions (e.g., from flight simulator): Fuzzy, Bayesian
  - Natural language: Use LSA, keywords, Atlas...
- ◆ Recognizing pedagogically important student errors
- ◆ Managing the student-tutor dialogue
  - Immediate feedback + hint sequences
  - Delayed feedback + student or tutor controlled review
  - Adaptive, especially decision theoretic & fading

## 4 popular designs for the Expert



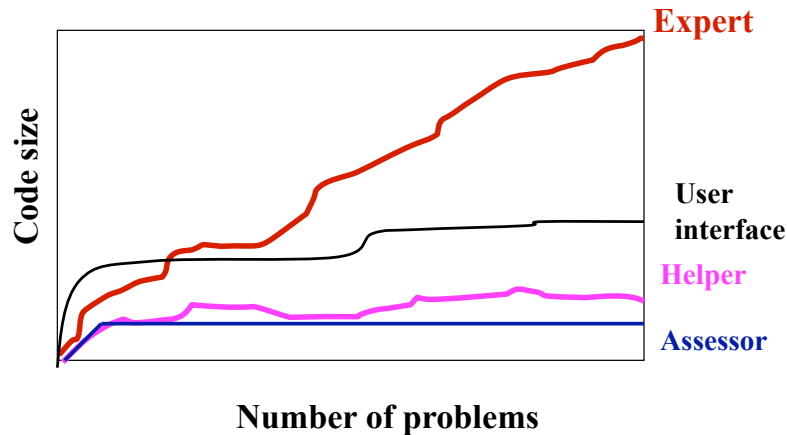
- ◆ Hand-author all possible solutions per problem
  - AutoTutor, CTAT (sort of; formulas can express wide range of solutions)
- ◆ Rule-based AI problem solver + problem  $\rightarrow$  all possible solutions
  - Andes, Cognitive tutors
- ◆ Hand-author one solution & use constraints to generalize to all possible solutions
  - Constraint-based tutors e.g., SQL Tutor
- ◆ Hand-author a few complete solution graphs, then machine-learn the rule-based AI problem solver
  - Steve; PSLC SimStudent

## Assessor: Main design issues



- ◆ Granularity
  - Knowledge components?
  - Unit mastery?
  - Overall competence?
- ◆ Interpretation of hints and errors in response patterns
- ◆ An incorrect step is not recognized, which correct step was the student trying to enter?
  - Can use step's location (Cognitive tutors; SQL)
  - Can use temporal sequence (Steve)
- ◆ Assignment of credit & blame when multiple knowledge components per step

## Where's the scale-up bottleneck?



## Other scaling up issues = same as other reforms

- ◆ Coordination with curriculum & standards
- ◆ Teacher buy-in and training
- ◆ Support
- ◆ Etc...

## Implementation: Summary

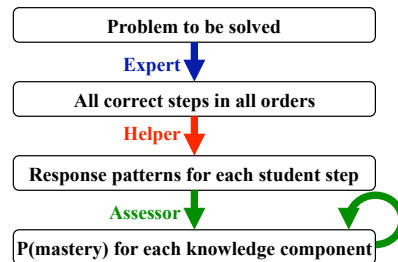


### ◆ Four main modules

- Expert
- Helper
- User interface
- Assessor

### ◆ Scaling up issues

- Novel: Code grows with number of problems
- General: Integration, buy-in, training, support...



## Questions?



### ◆ Outer loop

- How to select the next task
- How to obtain a large set of tasks

Next

### ◆ Inner loop

- Minimal feedback on a step
- Hints on the next step
- Error-specific feedback on an incorrect step
- Assessment of student's knowledge.
- Review of the student's solution

### ◆ Implementation and scale-up