moving lives forward

Design and Pilot Testing of Subgoal Labeled Worked Examples for Five Core Concepts in CS1

Briana B. Morrison

Lauren Margulieux, Adrienne Decker

UNIVERSITY OF NEBRASKA OMAHA

Georgia State University | COLLEGE OF EDUCATION & HUMAN DEVELOPMENT

University at Buffalo

The Trouble with Transfer

- How we want to teach people
  - Abstract conception of procedure (how to write a loop)
  - Study worked examples (add up waiter's tips, average minutes of exercise)
  - Solve novel problems (average rainfall)
- How people want to learn
  - ~~Abstract conception of procedure~~ (Jonassen, 2000)
  - Study worked examples (LeFevre & Dixon, 1986) (add up waiter's tips)
  - Solve problems that are slightly different than examples (add up bartender's tips)

The Trouble with Transfer

- Novices organize information based on surface features of problems (Chi, Feltovich, & Glaser, 1981)
- Knowledge structures based on surface features obscure what learners consider to be similar problems (Reed, Ernst, & Banerji, 1974)
- The lesson: Studying worked examples both helps students grasp concepts but also inhibits transfer (Eiriksdottir & Catrambone, 2011)

Subgoal Learning

- Subgoals = Functional pieces of a problem solution
- Example: Solve for x

4x - 8 = 2x + 6	
+ 8 = + 8	Isolate variable
- 2x = - 2x	
4x - 2x = 6 + 8	
2x = 14	Simplify terms
/2 = /2	
x = 7	

Subgoal Labels Effectiveness

- Subgoal labeled worked examples improve performance for
  - Block-based programming

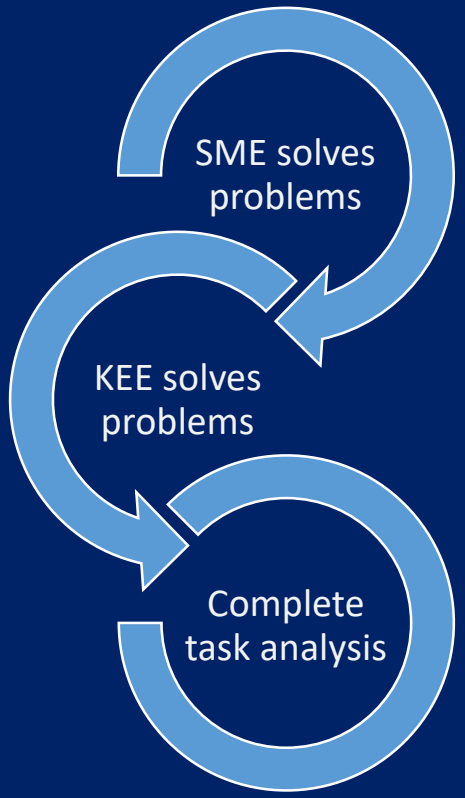
    Margulieux, Guzdial, & Catrambone, 2012; Margulieux & Catrambone, 2016; Margulieux, Catrambone, & Guzdial 2016
  - Text-based programming

    Morrison, Margulieux, & Guzdial, 2015; Morrison, Margulieux, Ericson, & Guzdial, 2016; Morrison, Decker, & Margulieux, 2016
  - K-12 Teachers

    Margulieux, Catrambone, & Guzdial, 2013

Task Analysis by Problem Solving

- TAPS protocol
  - Subject matter expert (SME)
  - Knowledge extraction expert (KEE)
  - Focus on problem solving, not teaching
  - Identify areas of tacit knowledge



Sample Subgoal Labels

Subgoals for evaluating and writing selection statements

A. Evaluate selection statement

- 1. Diagram which statements go together
- 2. For if statement, determine whether expression is true or false
- 3. If true – follow true branch, if false –follow else branch or do nothing if no else branch

B. Write selection statement

- 1. Define how many mutually exclusive paths are needed
- 2. Order from most restrictive/selective group to least restrictive
- 3. Write if statement with Boolean expression
- 4. Follow with true bracket including action
- 5. Follow with else bracket
- 6. Repeat until all groups and actions are accounted for

List of Topics We TAPS-ed

- Expression statements
- Selection statements
- Loops
- Arrays
- Object instantiation and method calls
- Writing classes

- Designed worked example – practice problem pairs

Pilot Study

- Compare groups at UNO (*N* = 307) Fall 2018
  - Received traditional worked examples and practice problems
  - Received subgoal labeled worked examples and practice problems
- Everything else was the same
  - Qualifications of instructors
  - TAs
  - Quizzes (collected data)
  - Exams (collected data)
  - Labs
  - Assignments

Results - Quizzes

Total score = total points earned, including zeros

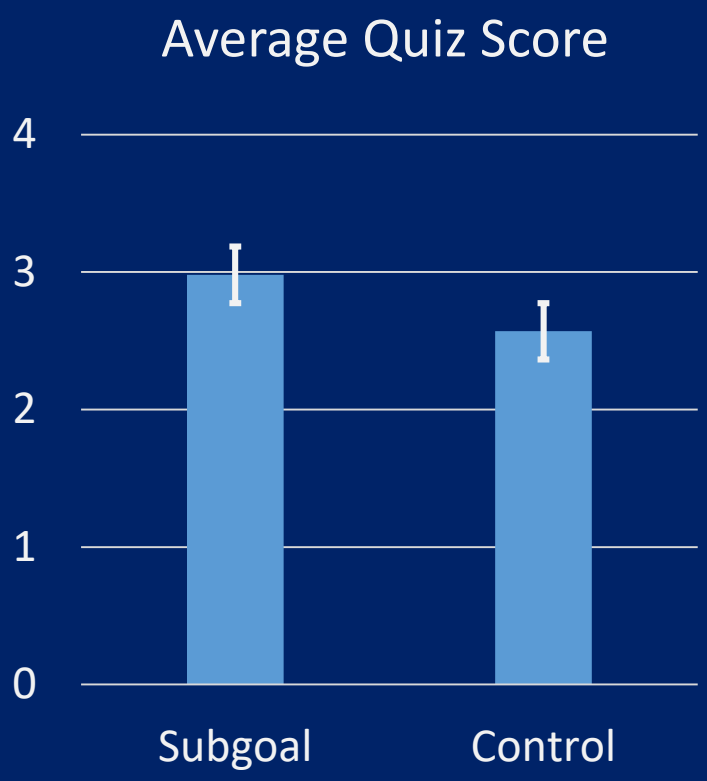
Average score = average of quizzes or exams taken, excluding zeros

Quizzes:

Total score higher for subgoal group, *d* = 0.42

Average score higher for subgoal group, *d* = 0.44

Consistent effect across topics



Results - Exams

Total score = total points earned, including zeros

Average score = average of quizzes or exams taken, excluding zeros

Exams:

Total score higher for subgoal group, *d* = 0.26

Average score not different for subgoal group, *d* = 0.20

- Substantial sample size, so not under-powered
- Benefits of subgoal labels likely diminished with more learning
- Subgoal group more likely to persist to final exam
- Subgoal group also consistently had lower variance (fewer students performing below a passing grade)

Limitations

- Conditions relatively good for natural experiment
- However, the researcher was the instructor for the course with subgoals
  - Necessary for first implementation of instructional materials
  - More research in varied settings is needed to determine generalizability

Takeaways

- Subgoal learning can improve problem solving in CS1
- Effects of subgoal learning diminish with more learning within a topic, but not for later topics in the course
- Subgoal learning can help students who would typically perform poorly in CS1
- Subgoal learning can improve persistence in CS1

Thank you!
Questions?

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 1927906 and 1712231. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Georgia State University

UNIVERSITY OF NEBRASKA OMAHA

University at Buffalo

Subgoals identified through TAPS protocol

Subgoals for evaluating and writing expression (assignment) statements	
A. Evaluate expression statement	B. Write expression statement
1. Determine whether data type of expression is compatible with data type of variable 2. Update variable for pre based on side effect 3. Solve arithmetic equation 4. Check data type of copied value against data type of variable 5. Update variable for post based on side effect	1. Determine expression that will yield variable 2. Determine data type and name of variable and data type of expression 3. Determine arithmetic equation with operators 4. Determine expression components 5. Operators and operands must be compatible
Subgoals for evaluating and writing selection statements	
A. Evaluate selection statement	B. Write selection statement
1. Diagram which statements go together 2. For if statement, determine whether expression is true or false 3. If true – follow true branch, if false –follow else branch or do nothing if no else branch	1. Define how many mutually exclusive paths are needed 2. Order from most restrictive/selective group to least restrictive 3. Write if statement with Boolean expression 4. Follow with true bracket including action 5. Follow with else bracket 6. Repeat until all groups and actions are accounted for
Subgoals for evaluating and writing loops.	
A. Evaluate loops	B. Write loops
1. Identify loop parts a. Determine start condition b. Determine update condition c. Determine termination condition d. Determine body that is repeated 2. Trace the loop a. For every iteration of loop, write down values	1. Determine purpose of loop a. Pick a loop structure (while, for, do_while) 2. Define and initialize variables 3. Determine termination condition a. Invert termination condition to continuation condition 4. Write loop body a. Update loop control variable to reach termination
Subgoals for calling and writing methods	
A. Call or trace method calls	B. Write methods
1. Classify method as `static` method or `instance` method a. If `static`, use the class name b. If `instance`, must have or create an instance 2. Write (instance / class) dot method name and ( ) 3. Determine whether parameter(s) are appropriate a. Number of parameters passed must match method declaration b. Data types of parameters passed must match method declaration (or be assignable) 4. Determine what the method will return (if anything: data type, void, print, change state of object) and where it will be stored (nowhere, somewhere) 5. Evaluate right hand side of assignment (if there is one). Value is dependent on method's purpose	1. Define method header based on problem 2. Define return statement at the end 3. Define method body/logic a. Determine types of logic (expression, selection, loop, etc.) b. Define internal variables c. Write statements
Subgoals for evaluating and writing arrays	
A. Evaluate arrays	B. Write arrays
1. Set up array from 0 to size-1 2. Evaluate data type of statements against array 3. Trace statements, updating slots as you go a. Remember assignment subgoals	1. Data type plus [ ] 2. Variable name = `{initializer list}`, or `new datatype [size]`