8-2019

Using the SOLO Taxonomy to Understand Subgoal Labels Effect in CS1

Adrienne Decker
SUNY University at Buffalo, adrienne@buffalo.edu

Lauren Margulieux
Georgia State University, lmargulieux@gsu.edu

Briana B. Morrison
University of Nebraska at Omaha, bbmorrison@unomaha.edu

Recommended Citation
the SOLO Taxonomy to Understand Subgoal Labels Effect in CS1. In Proceedings of the Fifteenth Annual Conference on International Computing Education Research. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3291279.3339405

Using the SOLO Taxonomy to Understand Subgoal Labels Effect in CS1

Adrienne Decker
Engineering Education
University at Buffalo
Buffalo, NY USA
adrienne@buffalo.edu

Lauren E. Margulieux
Learning Sciences
Georgia State University
Atlanta, GA USA
lmargulieux@gsu.edu

Briana B. Morrison
Computer Science
University of Nebraska Omaha
Omaha, NE USA
bbmorrison@unomaha.edu

ABSTRACT

This work extends previous research on subgoal labeled instructions by examining their effect across a semester-long, Java-based CS1 course. Across four quizzes, students were asked to explain in plain English the process that they would use to solve a programming problem. In this mixed methods study, we used the SOLO taxonomy to categorize student responses about problem-solving processes and compare students who learned with subgoal labels to those who did not. The use of the SOLO taxonomy classification allows us to look deeper than the mere correctness of answers to focus on the quality of the answers produced in terms of completeness of relevant concepts and explanation of relationships among concepts. Students who learned with subgoals produced higher-rated answers in terms of complexity and quality on three of four quizzes. Also, they were three times more likely to discuss issues of data type on a question about assignments and expressions than students who did not learn with subgoal labeling. This suggests that the use of subgoal labeling enabled students to gain a deeper and more complex understanding of the material presented in the course.

CCS CONCEPTS

• **Social and professional topics** → Computing education → *Computer science education* → *CS1*

KEYWORDS

CS1, subgoal labeling, SOLO taxonomy, introductory programming

ACM Reference format:

Adrienne Decker, Lauren E. Margulieux, Briana B. Morrison. 2019. Using the SOLO Taxonomy to Understand Subgoal Labels Effect in CS1. In *Proceedings of the Fifteenth Annual Conference on International Computing Education Research. ACM, New York, NY, USA, 9 pages.* http://dx.doi.org/10.1145/3291279.3339405

ICER '19, August 12–14, 2019, Toronto, Ontario, Canada.

ACM ISBN 978-1-4503-6185-9/19/08.
DOI: http://dx.doi.org/10.1145/3291279.3339405

1 Introduction

Subgoal-labeled worked examples (SLWE) have shown promise in tackling the persistent problems of low retention and success in introductory programming courses at the college level [13–15]. However, these previous studies have exposed students to subgoals for only one to two hours of instructional time. The current project extends this work by exploring the use of subgoal-labeled worked examples throughout an entire introductory Java programming course [12]. The materials were pilot tested from August to December 2018 to examine their effect on student performance. This paper discusses the analysis of the data collected on quiz questions that instructed students to *explain in plain English* the process that they would use to solve a programming problem. Student answers were scored using the SOLO taxonomy, which represents answer complexity and completeness. The guiding research question for this work was: *If students learn procedures using SLWE, do they create more complex and complete answers to explain in plain English questions than students who learn using non-subgoal-oriented materials?*

2 Background

2.1 Subgoal Learning

Subgoal learning explicitly teaches students the subgoals, or functional pieces, of a problem-solving procedure. For example, to solve a problem with a `while` loop, students must determine a stopping case for the loop, so defining a termination condition is a subgoal of solving a problem with a `while` loop. The specific steps taken to achieve this subgoal varies from problem to problem, but the function remains the same. Novices solve programming problems better when they explicitly learn the subgoals of a procedure because they often do not recognize these functional pieces on their own [3, 4, 8, 13-15].

Students typically learn subgoal through subgoal labeled worked examples. Worked examples are commonly used to teach well-structured problem-solving procedures because they demonstrate how to apply an abstract procedure to a concrete problem before the learner can solve problems independently [1, 19, 23]. The drawback of worked examples, however, is that they must include details specific to a problem. For example, to demonstrate how to solve a problem using a `for loop`, the worked example also includes a context, such as "write a loop that

calculates the average age of the first 100 people to take a survey." Learners tend to organize information about the procedure using these easy-to-grasp details, like age, rather than around the hard-to-conceptualize abstract procedure that they are learning, leading to difficulty transferring knowledge to new problems [1, 18]. Subgoal learning addresses this problem by pointing out shared functional features in worked examples, helping learners to organize information so that it can transfer more easily [4, 13].

2.2 SOLO Taxonomy

The Structure of the Observed Learning Outcome (SOLO) taxonomy was introduced by Biggs and Collis [2] to provide a framework for more consistent, qualitative evaluation of student responses to open-ended questions. The taxonomy was designed based on student responses to open-ended questions in multiple disciplines. The taxonomy has three dimensions:

1. Capacity: the pieces of information used to produce the response, ranging from low (i.e., only the information in the question and one relevant piece of information) to high (i.e., the question, multiple pieces of relevant information, interrelations among information, and abstract principles)
2. Relating operations: the relationship between the question and response, ranging from illogical (e.g., tautologies), to question-specific information only (i.e., answers the question without relating to principles or concepts), to information that generalizes beyond the specific question (i.e., relating response to abstract principles and concepts)
3. Consistency and closure: the consistency between information provided and the conclusion that the student comes to, ranging from not answering the question, providing inconsistent evidence or jumping to conclusions, to consistent evidence and multiple conclusions based on relevant possible alternatives

Using the three dimensions, Biggs and Collis defined five levels of structural complexity, which can be used to determine how well students learned an objective. Students demonstrate their knowledge of the subject at one of the five levels of complexity:

- Prestructural: little to no understanding of the topic
- Unistructural: understanding of a single aspect of the topic
- Multistructural: understanding of several aspects of the task but each aspect is represented independently
- Relational: understanding of several aspects of the task and how they are related
- Extended Abstract: understanding of the aspects can be generalized beyond the context of the question

Based on their analysis of student responses, complexity is typically at the same level across the three dimensions. For example, a prestructural response will typically match the prestructural criteria in 1) capacity, 2) relating operations, and 3) consistency and closure. Occasionally, a transitional answer will exist between two levels of dimensions.

2.3.1 SOLO in Computing Education Research. The SOLO taxonomy has been used extensively in computing education research. A 2004 ITiCSE Working Group (the Leeds Group) [10] provided the first attempt at mapping the SOLO taxonomy to computing. Table 1 summarizes this initial mapping, which is most used by other studies, including ours.

Category	Definition
Prestructural	Significant misconception or preconception irrelevant to programming
Unistructural	Correct grasp of some but not all aspects of the problem (i.e., educated guess)
Multistructural	Understands all parts of the problem but does not exhibit an awareness of the relationships between the parts; the answer may be correct or not
Relational	Parts of the problem are integrated into a structure; the answer may be correct or not
Extended	The response goes beyond the immediate problem and links to a broader context

Table 1. Mapping of SOLO taxonomy to computing [10].

The SOLO taxonomy, along with the *explain in plain English* (EiPE) questions, have been used many times within computing education research, especially for CS1. The BRACElet project studied the relationship between novice programmers' code writing ability and their explanations of code [11, 24]. In 2011, Corney et al. [5] explored student EiPE responses for swapping variables, code that represents the simplest case in which a programming student can manifest a SOLO relational response. Sheard et al. [21] studied exams for CS1 students. They found that reading tasks correlated positively with performance on writing tasks and that undergraduate students had a lower SOLO score than postgraduate students.

Others have modified the SOLO taxonomy to better map to the concepts they were measuring. In Sudol-DeLyser [22], a modification was done to the SOLO classification scheme to capture the number and types of abstraction statements made by students during a think-aloud protocol. Results indicate that students with greater proficiency at writing code were more likely to use multiple levels of abstraction when describing the code they were writing and moved between levels of abstraction with higher frequency. Izu et al. [7] used an adjusted SOLO taxonomy to classify programming questions' by using a "building block" as the granular structure in the taxonomy to overcome the variability in problem difficulty. A building block was defined as a code pattern or template that students had seen, allowing for differentiation between recall and synthesis in problem difficulty. Murphy et al. [16, 17] replicated the Leeds Working Group results while using Table 1's categories but without the Extended classification. Their results support a relationship between explaining and writing code. Beyond CS1, Corney et al. [6] used SOLO for CS2 Data Structures questions (again with no Extended Abstract category). The results found a strong correlation between students' ability to explain code at an abstract level and performance on code writing and code reading test problems at this level.

3 Present Study

We tested the SLWE in introductory programming courses at a medium-sized Midwestern university in the United States. The university offered five sections of a Java-based CS1 course from

August to December 2018. Students were free to enroll in any lecture or lab section. The lecture sections were taught by three full-time faculty, and all had more than 15 years of experience teaching introductory programming. Two sections were taught by one of the authors and incorporated SLWEs in place of the conventional worked examples used in the other three sections. All sections were coordinated and used the same textbook, slides, peer instruction questions, pace of topics, quizzes, tests, labs, and Learning Management System (LMS) instance. The only difference between the sections was the examples used for in-class practice and the introduction of the subgoals. The subgoals used in class are given in [12]. The intervention sections used the developed SLWE and practice problems while the other sections used instructor developed examples.

The present study compares student responses to four EiPE questions between the sections that used SLWE (i.e., subgoal group) and those that used the conventional examples (i.e., control group). One of the three control sections was taught online and initially treated as a separate group in case students who chose to enroll in the online course were different than the other students in some way. After analyzing the data, however, the online section was indistinguishable from the other control sections. Thus, the three sections are treated as one control group in the analysis.

3.1 Previous Results

In a previous paper [12], we presented the analysis that compares quiz and exam grades by group, which we summarize here to contextualize the EiPE responses that are the focus of the present paper. We had 120 students in the subgoal group and 145 students in the control group. Several learner and demographic characteristics of the students were collected, but none of them correlated with group or grades. We examined student grades on the five quizzes given after SLWE were used (out of 15 weekly quizzes) and all four exams, each of which included multiple choice and short answer questions that were automatically graded. The exams also included long answer questions, which were graded by the same person across all sections.

The quiz and exam grades were analyzed in a few ways to gain a complete understanding of the data. First, an *average* score was calculated for each student. This score represents the average grade on quizzes or exams that the student submitted. Any missing grade was not included in the average score. Second, a *total* score was calculated for each student. This score included all available points for quizzes or exams, and if a student did not turn in a quiz or exam, it was treated as a zero. The total score was paired with the *number* of quizzes and exams completed to help us to consider the role of the SLWE on the dropout rate.

On the five quizzes, students in the subgoal group performed better than students in the control group with a medium effect size, $d = 0.42$ for the average score and $d = 0.44$ for the total score. In addition, students in the subgoal group completed more quizzes that students in the control group. An interesting result from the analysis was that the subgoal group has a significantly lower variance in scores than the control group. Given that the subgoal group also performed better than the control group, it could be the case that the subgoals particularly improved the grades of students who would have performed poorly on the quizzes. In [12], we argued that this pattern of results suggests that at-risk students were less likely to drop out of the subgoal sections than control sections of the course.

On the four exams, students in the subgoal group performed better than students in the control group only on the total score (i.e., including zeros for missing exams). On the average exam score, the groups performed equivalently, but the subgoal group again had a lower variance in scores than the control group. Based on the number of exams taken in each group, the difference between groups in the total score is likely due to the zeros from missing exams. In [12], we argued this pattern of results again suggests that students in the subgoal section were less likely to drop out of the course. For students who persisted through the course, SLWE did not improve the average score on exams, though it did reduce variance, which again may point to helping at-risk students.

SLWE improving quiz scores, but not exam scores, aligns with the subgoal learning framework, which is designed to help novices understand the structure of problem-solving procedures before they are able to recognize it for themselves. By the time a student has studied for an exam, they are likely to be able to recognize problem-solving structures. Therefore, it is expected the SLWE have a stronger effect on the quiz grades, which represent initial knowledge, than exam grades, which represent well-developed knowledge. For the present study, we explore how SLWE affect students' initial conceptions of problem-solving procedures through EiPE questions given on the quizzes.

4 Methods

The total number of students across all sections was 307 based on enrollment at the beginning of the semester. Students were excluded from analysis if they did not complete at least one exam or one quiz, effectively withdrawing from the course. The final sample size was $N = 265$, 145 in the control group and 120 in the subgoal group--the same sample as used in the analysis for [12].

4.1 Data Collection Sources

Student performance on four quizzes was collected. Below are the characteristics of the student performance items:

- The quiz questions were short answer, specifically questions that instructed students to *explain in plain English* about code. See Figure 1 for two example questions.
- Each quiz question analyzed here was worth only one point on a 5- to 15-point quiz and, even cumulatively, had almost no effect on students' course grades.
- Questions of this type were included on 4 of 15 weekly quizzes. (Administered during weeks 4, 8, 10, and 12 of the term.) Explanations of the topical coverage of the questions are presented by quiz in Section 5.
- Neither the subgoal group nor the control group practiced this type of question during class.
- Quizzes were assigned from Friday morning until midnight Monday at midnight with a 20-minute time limit and

completed online through the LMS.

- The EiPE quiz questions were graded so that any reasonable answer was given full credit.
- Students were not given feedback on their responses to the quiz questions analyzed here.

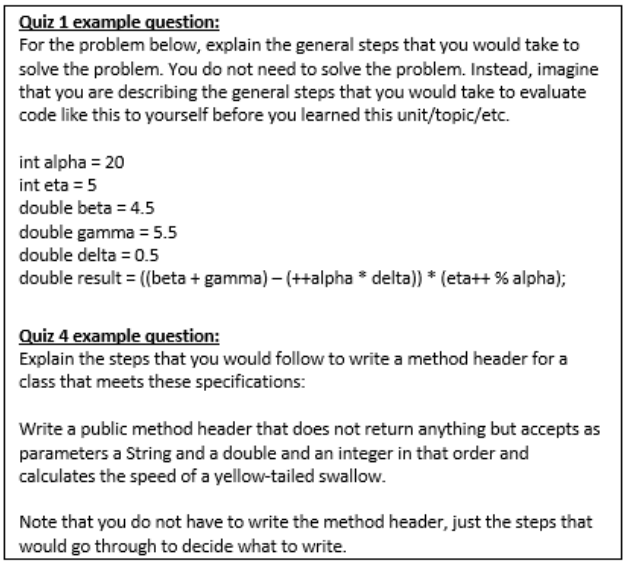

Figure 1. Sample EiPE questions from quizzes.

4.2 Classifying Responses Using SOLO

For each quiz, anonymized student responses were coded based on the SOLO taxonomy categories: prestructural, unistructural, multistructural, relational, or extended abstract. The anonymous student responses were graded as one set, with no indication of whether they were in the subgoal or control group. The process for coding the responses involved three coders working concurrently on the coding process. To start the process, the first several responses (about 10) for each question were coded cooperatively by all three coders discussing where each response fell into the taxonomy and why each believed that categorization to be correct. Discrepancies were discussed until agreement was reached on the code and a general understanding of what was expected in each response for each category was reached.

Each of the quiz questions had specific concepts, relationships, and principles that we used to distinguish between levels of the SOLO taxonomy. The coding rubrics were related to the information included in the responses rather than to subgoal labels. For example, for the first quiz, whether students discussed matching data types between the right and left sides of an expression statement could be the distinction between a three or a four rating. The question-specific distinctions are included in the results section to help contextualize the findings.

After the initial 10 responses, the three coders worked independently on an additional set of 10 responses and compared answers after they had scored the set and resolved differences. This process continued until 20% of the responses were coded by all three raters. If an acceptable level of interrater reliability was reached, the coders divided up the remaining responses and coded independently. If interrater reliability was not acceptable, they coded an additional 20% of responses until reliability was above the threshold.

To evaluate interrater reliability, we used the intraclass correlation coefficient of absolute agreement, ICC(A). We chose this test of reliability because it determines whether multiple raters give the same score to different student responses. Other more popular tests of reliability determine whether raters are consistent in ranking across responses. For example, if one rater gave consistently low scores to responses and another gave consistently high scores to responses, they could still achieve high interrater reliability with Cohen's kappa or intraclass correlation coefficient of consistency. However, for the SOLO taxonomy, we need to determine how often raters give the same score because each score has a qualitatively unique meaning. An ICC(A) value of 0.75 or higher is considered good interrater reliability [9].

5 Results

The SOLO taxonomy categorizes students' responses based on qualitative differences. Therefore, even though we have assigned numeric values to student responses, it is not necessarily appropriate to use those values mathematically. The SOLO categories yield ordinal data, which means that the categories are rank ordered (e.g., a five is better than a four) but the difference between values is not mathematically equal (e.g. the difference between a five and a four is not necessarily the same as the difference between a four and a three). Because we used ordinal data, the results of any parametric test (i.e., ANOVA) should be interpreted with extreme caution. Technically, using ordinal data violates the assumptions of a parametric test. We decided to present the results of these tests because the tests for homogeneity of variance, which helps determine the appropriateness of the data for parametric tests, were non-significant. The non-significant results for Mauchley's (for repeated measures) and Levene's (for t-test) tests suggest that the variance of data was normally distributed and equivalent between groups, which are the major concerns for analyzing ordinal data parametricly. We used the parametric tests to provide only a high-level view of the data before providing more nuanced, reliable descriptive statistics.

To explore the quality of students' responses across quizzes, we used a repeated measures analysis that links students' scores on each quiz question. Repeated measures analysis requires a data point for each measurement, and due to missing data on some quizzes, the sample size with complete data was limited, subgoal n = 53 (44% of the total sample) and control n = 44 (30% of the sample). Even with a limited sample size, the analysis found that the quiz question was a strong predictor of students' scores, F = 29.9 (sphericity was not violated, p = .215, so no correction was used), p < .01, partial η^2 = .24. This result suggests that the effect of quiz questions would have overshadowed any effect of learner differences (i.e., within-subjects variance), such as whether a student was more likely to give five or four rated responses. To further support this finding, we visually inspected all scores of students who received a five (i.e., extended abstract) on one of the quizzes. Their scores on the other responses follow a normal

distribution with the most common score on other quizzes being a 3. This pattern of results suggests that students' scores, even for students who received a five, were more affected by the question being asked than by personal characteristics.

Because the quiz score was a large predictor of performance, the subgoal and control groups were compared for each quiz independently. For each quiz, we used a t-test to compare groups, but as stated earlier these results are of limited usefulness. To explore the data, we examine the mode and frequencies of each score. We use these statistics instead of the typical mean and standard deviation because it more accurately describes ordinal data. For example, the standard deviation in our data was always around 1.0 because most students scored a three or four, and the numerical difference between those values is 1.0. Thus, this value is an artifact of the data analysis method rather than a meaningful representation of the variance in groups.

5.1 Quiz 1 – Expression Statements

In the first quiz analyzed (quiz 4 for the term), we had responses from 84 students in the subgoal group and 75 students in the control group. The raters reached a high level of interrater reliability, ICC(A) = 0.85, for the first 20% of responses. Because this quiz was early in the semester, the amount of knowledge that students could demonstrate was limited. This question asked students to explain the steps needed to evaluate a complex arithmetic expression that included both parenthesized sub-expressions and pre- and post- increment/decrement operators (see Figure 1). Table 3 provides the question-specific content, relationship, and principle information required for each score, i.e. the rubric.

Overall, the subgoal group had higher SOLO ratings than the control group on the first quiz, $t(158) = 19.14$, $p < .01$, $\eta^2 = .11$. To examine the differences between groups, we considered the mode and frequencies of each score in the groups (see Table 2). Based on frequencies, 68% of the students in the subgoal group were able to write answers to this question at either a four or a five rating, indicating that the students explained how to solve the arithmetic expression and how to deal with pre/post increment operations and, in many cases, data type compatibility.

In the control group, students achieved these rankings at roughly half of this rate (37%). Only 11% of the students in the subgoal group did not articulate the steps needed to solve the basic parts of the arithmetic expression (i.e., scores of one or two), while the control group had over twice that rate (27%). Overall, the subgoals seemed to enable students to articulate more information about the process of evaluating arithmetic expressions.

Note that the criteria to receive a four or five explicitly involves the mention of data types or type compatibility. This issue is explicit in the subgoals for this part of the course. Specifically, one of the subgoals states, "Determine whether the data type of expression is compatible with the data type of variable". As such, we undertook another analysis to see how often students in the two groups mentioned the issue of data types in their responses. The rate is almost double for the subgoal group (29%) than the control group (12%) for students mentioned compatibility in their answers. We expect that this is due to the explicit subgoal that was used in the course dealing with type compatibility drawing students' attention to the importance of data types.

	1	2	3	4	5
Subgoal Mode = 4	1 (1%)	8 (10%)	18 (21%)	43 (51%)	14 (17%)
Control Mode = 3	6 (8%)	14 (19%)	27 (36%)	25 (33%)	3 (4%)

Table 2. Quiz 1 score frequencies between subgoal and control groups.

SOLO	Description	Example
1 – prestructural	Nonsensical answer or answer that had no more information than the question provided	"Solve each equation."
2 – unistructural	Described how to solve part of the problem, but the description was incomplete	"First I would do the things within each set of parentheses. Second, I would do the multiplication. Finally I would subtract."
3 – multistructural	Described how to solve the complete problem but provided no explanation beyond the question at hand	"You need to follow the order of precedence for Java, so first you would do what is in the parentheses. In the parentheses you would do the ++ first from right to left, followed by modulus, then multiplication and division from left to right."
4 – relational	Described how to solve the problem and explained in abstract terms either how to evaluate pre- and post-increments or how to evaluate the appropriateness of data type between the variables	"First I would take the values within the parentheses and try to solve for those first. Starting with the one that has multiplication first, then modulus, and last, addition. ++Alpha would need 1 added to its value since it is a pre added value. Eta++ would add 1 to its value after solving for the result then take the modulus of eta++ % alpha."
5 – extended abstract	Described how to solve the problem and explained how to evaluate data type and increments for expression statements in general	"First thing I like to establish is what is an int, what is a double, and then what kind of answer do they want. We know they are looking for a decimal because it is a double. Next, go to the equation and treat it like math class using the orders of operation; PEMDAS. Starting from the beginning of that rule we have parenthesis, so we'll start by doing everything within their respected parenthesis. beta + gamma is pretty general, just add the two together. ++alpha * delta you want to add one to the variable alpha and then multiply that with delta. eta++ %alpha you will start by doing eta modular alpha and then add 1 because the ++ comes after the effected variable. Now follow order of operations."

Table 3. SOLO Categories for Quiz 1, Expression Statements.

5.2 Quiz 2 – Loops

In the second quiz analyzed (quiz 8 for the term), we had responses from 98 students in the subgoal group and 97 students in the control group. The raters reached a moderate level of interrater reliability, ICC(A) = 0.72, for the first 20% of responses. Thus, the raters discussed the criteria and rated an additional 20% of responses together to reach a high level of reliability, ICC(A) = 0.82. The remaining responses were scored by one rater. This question asked students to explain the steps needed to write code for a process that involved a single loop that processed input from the user and accumulated a sum. Table 5 provides the question-specific information required for each score.

Overall, the subgroup group scored higher than the control group on the second quiz , $t(194) = 11.62$, $p < .01$, $\eta^2 = .06$. We examined the differences between groups with the mode and frequencies of each score in the groups (see Table 4). This question asked about the code that would need to be written to solve the presented problem. We see a high occurrence of threes in this data (50% for subgoals, 46% for control). The students most often explained what needed to be done to solve the problem but showed no evidence of abstract thinking. For students who crossed into the relational category the rate was higher for the subgoal group (29% vs. 17%) and the subgoal group had the only five for this question. Also, there is a higher proportion of proportion of students who were not able to give a complete explanation of an answer in the control group versus the subgoal group (37% vs. 18% at a rating of one or two). This pattern also shows that in the control group, over one-third of the students could not give all the pieces required for an answer that earned a three score. We expect that the reason for the differences in this question was that the subgoals for loops gave the students a place to start their explanation and a way to articulate the pieces of the answer even if it was not at the higher cognitive levels of relational or extended abstract.

	1	2	3	4	5
Subgoal Mode = 3	5 (5%)	13 (13%)	49 (50%)	28 (29%)	3 (3%)
Control Mode = 3	11 (11%)	25 (26%)	45 (46%)	16 (17%)	0 (0%)

Table 4. Quiz 2 score frequencies between subgoal and control groups.

5.3 Quiz 4 – Writing Methods

We discuss the fourth quiz (quiz 12 for the term) before the third quiz because the first, second, and fourth quizzes follow the same pattern of results, and the third quiz does not. In the fourth quiz, we had responses from 92 students in the subgoal group and 92 students in the control group. The raters reached a high level of interrater reliability, ICC(A) = 0.87, for the first 20% of responses, and the remaining responses were scored by one rater. The question asked students to explain the steps needed to write a method header for a described method. Table 6 provides the question-specific information required to achieve each score.

The subgoal group scored higher than the control group on the fourth quiz, $t(183) = 25.08$, $p < .01$, $\eta 2 = .12$. We also examined group differences between mode and frequencies (see Table 7).

SOLO	Description	Example
1 – prestructural	Nonsensical answer, an answer that had no more information than the question provided, or alluded to a relevant principle but not in enough detail to apply it to the problem	"You would get the score for 9 hole for each round then print each round out."
2 – unistructural	Described 1-2 concepts that applied to the problem, but description was incomplete	"Declare variables for 18 holes. Println asking for input for each hole, using scanner. Println the sum of holes 1-9. Println the sum of holes 10-18. Print ln the sum of all holes."
3 – Multistructural	Described all concepts needed to solve the problem, whether they were correctly applied, but provided no explanation beyond the question at hand	"Have the scanner along with the 18 variables needed to add up golf scores. After the user inputs all of the numbers then you can have the system add them all up and print it. Then if you want the sum of the 1st nine holes then add up the 1st nine variables and the same for the 2nd half."
4 – relational	Described how to solve the problem and explained how the different pieces of the solution related to each other	"I would make two loops that would count up to 9 times for each side of the golf course. Then within the loop I would have the hole score added to the total score as well as add a counter for that hole. This would be the exact same for both sides of the course and at the end of each loop I would print out the total score for those loops then I would add the two scores together to get a total score for the 18 holes."
5 – extended abstract	Explained how to solve a problem like this in abstract terms	"The first thing to do is to determine what kind of loop to use. Since the counter value or number of iterations is known for the program, both a while or a for loop will work. However, a for loop is simpler and more concise to use. Since you know that each half of the game needs to be scored, and will be scored the same way, the same block of code can simply be used twice and the sum values at the end of each block can be assigned to different variables to delineate which is the first nine and the second nine holes. In the for-loop the counter needs to loop exactly 9 times, once for each hole. Within the for loop the code needs gather the score for that hole through a user query and add it to a sum variable, to get the total score for the entire 9 nine holes. The code is then repeated to get the sum for the second nine holes, and then both sums are added together to get the total for the round of golf."

Table 5. SOLO Categories for Quiz 2, Loops

SOLO	Description	Example
1 – prestructural	Nonsensical answer, an answer that had no more information than the question provided, or confused classes and methods	"By determining a constructor and instance then I would write the code using methods to create how fast a swallow travels."
2 – unistructural	Described 1-2 concepts that applied to the problem, but the description was incomplete or described a class instead of a method	"Need to figure out what items are ints, doubles, strings. Then from what you are making and if it needs to accessed or created you would make it public or private class."
3 – Multistructural	Described all concepts needed to solve the problem but provided no explanation beyond the question at hand	"Write public, then void, and then in parenthesis create 3 variables, a string, a double, and an int in that order."
4 – relational	Described how to solve the problem and explained how the different pieces of the choices made to solve this particular problem	"I would start of by specifying that it's public since that was requested and add void since it does not return anything and I would call the method speed since that is what it is calculating and in the parenthesis I would add "String s, double a, int b" since it requested it in that order."
5 – extended abstract	Explained how to solve a problem like this in abstract terms	"First you would choose whether people should have access to this or not. Public is yes, and private is no. Next find if you need to return something or not. Since you don't, you would use void. If you needed to return something you would use the data type (int, double, etc). Then you choose a name that fits what you are creating. For this I will just use "speed". Then you would put the parameters in to what they will be entering. It says "a String and a double and an integer in that order". So you put that in the () of your method."

Table 6. SOLO Categories for Quiz 4, Writing Methods.

	1	2	3	4	5
Subgoal Mode = 4	1 (1%)	4 (4%)	27 (29%)	53 (58%)	7 (8%)
Control Mode = 3	11 (12%)	17 (19%)	32 (35%)	27 (29%)	5 (5%)

Table 7. Quiz 4 score frequencies between subgoal and control groups.

For this question, 31% of the students in the control group were rated only a one or a two on this question, compared to the 5% of the subgoal group. On the other end of the scale, 66% of the subgoal group were rated a four or five on this question, with only 34% of the control group receiving the same score. We expect that the subgoal labels for writing methods and evaluating methods helped students to explain how they would choose the various parts of the method header and, therefore, earn a higher score. 95% of the subgoal students were able to provide a complete answer to this problem compared to 69% of the control students.

5.4 Quiz 3 – Nested Loops

The third quiz analyzed (quiz 10 for the term) does not follow the same pattern of results found for the other quizzes. The third quiz had responses from 90 students in the subgoal group and 102 students in the control group. The raters checked interrater reliability after scoring each quintile of the scores, but they never achieved a sufficiently high reliability to warrant a single rater. Therefore, two raters discussed and reached agreement for each of the responses. This question asked the students to look at a nested loop structure and describe how they would determine its output. We believe that the low interrater reliability for this question was due to the numerous pieces of content knowledge required to answer this question. It is also interesting to note that for this question, the mode was two for both groups, occurring at twice or more times the rate of the other scores. This mode indicates that responses at the higher levels of the taxonomy were not as prevalent in the data set and did not allow for exemplars and discussion which could have also led to the interrater reliability issues. Table 6 provides the question-specific content, relationship, and principle information required to achieve each score.

The two groups scored equivalently on the third quiz, $t(191) = 1.13$, $p = .29$, $\eta^2 = .01$. The mode and frequencies of each score in the groups can be found in Table 8. For this question, the subgoal group did not produce answers at higher levels of the taxonomy at a greater rate than the control group. In fact, only 12% of the subgoal group received ratings of four or five, while 19% of the control group received those ratings. The students in the subgoal group had a higher proportion of ratings of two (53% vs. 38%). Though we cannot be certain why this question displayed such different outcomes than the others or what aspects of this question made the results different, we expect that the high level of content knowledge required for the question played a significant role. In addition, the use of the nested loop did not match well onto the subgoal labels, providing no extra benefit for students who learned with SLWE.

	1	2	3	4	5
Subgoal Mode = 2	8 (9%)	48 (53%)	23 (26%)	8 (9%)	3 (3%)
Control Mode = 2	12 (12%)	39 (38%)	32 (31%)	17 (17%)	2 (2%)

Table 8. Quiz 3 score frequencies between subgoal and control groups.

6 Conclusion

Overall, the subgoal label group gave more complete answers, often including relational and abstract information, on three of the four quiz questions. Based on the SOLO taxonomy, subgoal students demonstrated a higher level of cognitive understanding of the underlying programming principles. For the one question

SOLO	Description	Example
1 – prestructural	Nonsensical answer, an answer that had no more information than the question provided, or identified content in the code with no explanation for how they functioned	"I would start from the first for statement and then continue to the next for statement using the previous values needed."
2 – unistructural	Described 1-2 concepts that applied to the problem, but description was incomplete or did not demonstrate an understanding of the code presented to them	"First see what loops are grouped together by brackets. The first loop is contained by the first and last brackets and the loops inside this one only contain the next line after the for loop statement. I would find the output for each smaller loop and then everything inside the first loop then display that value the number of times each loop specified."
3 – Multistructural	Described all concepts needed to solve the problem but provided no explanation beyond the question at hand	"To solve this, I would first block each for loop so I know what loop is connected to what action (and what actions also come right after a loop has concluded). Then, I would determine how many times each individual loop would run. After that it is merely a game of running through steps. Loop at the bottom goes a couple times, loop above it goes once. Repeat this until the loop above it is complete, then go to the initial loop. This repeats until the initial loop also is finished executing. That should give you your output."
4 – relational	Described how to solve the problem and explained how the different pieces of the choices made to solve this particular problem	"First, look at the large 'for loop'. This loop will execute ten times but inside this loop there are two nested loops. The first inner loop will execute 10 times on the first round of the large loop, nine times during the second time through the large loop and so on. The System.out.println( ); after the first nested loop separates the '*' with an empty line each time the x value changes. The second nested loop is similar but is not inside the first nested loop because the first nested loop does not use curly braces. It is similar to the first nested loop and will print ten octothorps the first round of the large loop, then nine the second round through."
5 – extended abstract	Explained how to solve a problem like this in abstract terms	"First, notice that in the first for loop, x counts down from ten to one before becoming false, meaning it will execute the internal code 10 times. Next, the first internal for loop counts to whatever x is for that specific loop, counting from one each time. This will produce an asterisk for each iteration. Because it will only loop the code on the line below it, after the loop is finished, it will create a line break and move on to the second for loop. The next for loop will start out at ten each time, and reduce until it has reached the x value of that loop. Once this loop is finished, another line break will occur and the x loop will move onto its next iteration. This will result in Alternating lines of stars and pounds, with the stars decreasing by one each time, and the pounds increasing by one each time."

Table 9. SOLO Categories for Quiz 3, Nested Loops.

in which this was not the case, we argue that the question required more pieces of content knowledge, making it more difficult to achieve higher ratings on SOLO. A majority of students tended to write enough to earn a unistructural rating, but they did not expand upon their responses beyond the complex structure required for that question (nested loops). In addition, the subgoal labels from the SLWE did not fit the problem, which likely contributed to the subgoal and control students performing equivalently.

There is still much work to be explored in this area with regards to the effect subgoal labels have on students' development of knowledge. Although this analysis shows promising results, the pilot test has significant limitations. The instructor who was teaching using SLWE was also part of the research team. At the phase of the development of the subgoal materials, this was necessary to fix any errors or overlooked details that would disrupt using the materials in class, but it also diminishes the validity of our results. The instructor is a veteran at teaching introductory programming and, thus, has significant prior experience, which helps to increase consistency of instruction and reduce bias. Some level of bias, however, is still likely to have been represented in the data.

Our next steps are to test the SLWE in courses taught by instructors not directly part of the research team and analyze the student performance on the quizzes and exams from those courses. The courses will be in a wide range of universities taught by various instructors, and we will collect data from students with a wide range of learner characteristics. Based on those results, we will have a much clearer picture of the impact of implementing subgoal materials across an entire course.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under grants 1712025, 1712231 and 1927906. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Atkinson, R.K. et al. 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research.* 70, 2 (2000), 181–214.

[2] Biggs, J.B. and Collis, K.F. 2014. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome).* Academic Press.

[3] Brown, N.C. and Wilson, G. 2018. Ten quick tips for teaching programming. *PLoS computational biology.* 14, 4 (2018).

[4] Catrambone, R. 1998. The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General.* 127, 4 (1998), 355.

[5] Corney, M. et al. 2011. Early Relational Reasoning and the Novice Programmer: Swapping as the "Hello World" of Relational Reasoning. 114, (2011), 10.

[6] Corney, M. et al. 2014. "Explain in Plain English" questions revisited: data structures problems. *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14* (Atlanta, Georgia, USA, 2014), 591–596.

[7] Izu, C. et al. 2016. A Study of Code Design Skills in Novice Programmers using the SOLO taxonomy. *Proceedings of the 2016 ACM Conference on International Computing Education Research - ICER '16* (Melbourne, VIC, Australia, 2016), 251–259.

[8] Joentausta, J. and Hellas, A. 2018. Subgoal Labeled Worked Examples in K-3 Education. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 616–621.

[9] Koo, T.K. and Li, M.Y. 2016. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*. 15, 2 (2016), 155–163.

[10] Lister, R. et al. 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin* (2004), 119–150.

[11] Lister, R. et al. 2006. Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy. *ACM SIGCSE Bulletin* (2006), 118–122.

[12] Margulieux, L.E. et al. 2019. Design and Pilot Testing of Subgoal Labeled Worked Examples for Five Core Concepts in CS1. *ITICSE'19: Innovation and Technology in Computer Science Education Proceedings* (Aberdeen, Scotland, Jul. 2019), 7.

[13] Margulieux, L.E. et al. 2012. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. *Proceedings of the ninth annual international conference on International computing education research* (2012), 71–78.

[14] Morrison, B.B. et al. 2016. Learning Loops: A Replication Study Illuminates Impact of HS Courses. *Proceedings of the 2016 ACM Conference on International Computing Education Research* (New York, NY, USA, 2016), 221–230.

[15] Morrison, B.B. et al. 2015. Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (New York, NY, USA, 2015), 21–29.

[16] Murphy, L. et al. 2012. Ability to "explain in plain English" linked to proficiency in computer-based programming. *Proceedings of the ninth annual international conference on International computing education research - ICER '12* (Auckland, New Zealand, 2012), 111.

[17] Murphy, L. 2012. "Explain in plain English" questions: implications for teaching. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. (2012), 6.

[18] Renkl, A. 1997. Learning from worked-out examples: A study on individual differences. *Cognitive science*. 21, 1 (1997), 1–29.

[19] Schwonke, R. et al. 2009. The worked-example effect: Not an artefact of lousy control conditions. *Computers in Human Behavior*. 25, 2 (2009), 258–266.

[20] Seiter, L. 2015. Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15* (Kansas City, Missouri, USA, 2015), 540–545.

[21] Sheard, J. et al. 2008. Going SOLO to Assess Novice Programmers. *ACM SIGCSE Bulletin*. 40, 3 (2008), 5.

[22] Sudol-DeLyser, L.A. 2015. Expression of Abstraction: Self Explanation in Code Production. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15* (Kansas City, Missouri, USA, 2015), 272–277.

[23] Sweller, J. 2006. The worked example effect and human cognition. *Learning and instruction*. (2006).

[24] Whalley, J.L. et al. 2006. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies. *Australiasian computer science communications*. (2006), 10.