2015

Subgoals, Context, and Worked Examples in Learning Computing Problem Solving

Briana B. Morrison
University of Nebraska at Omaha

Lauren Margulieux
Georgia State University

Mark Guzdial
Georgia Institute of Technology - Main Campus

Subgoals, Context, and Worked Examples in Learning Computing Problem Solving

Briana B. Morrison
School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA, 30332-0760
bmorrison@gatech.edu

Lauren E. Margulieux
School of Psychology
Georgia Institute of Technology
654 Cherry Street
Atlanta, GA, 30332-0170
l.marg@gatech.edu

Mark Guzdial
School of Interactive Computing
Georgia Institute of Technology
85 5th Street NW
Atlanta, GA, 30332-0760
guzdial@cc.gatech.edu

ABSTRACT
Recent empirical results suggest that the instructional material used to teach computing may actually overload students' cognitive abilities. Better designed materials may enhance learning by reducing unnecessary load. Subgoal labels have been shown to be effective at reducing the cognitive load during problem solving in both mathematics and science. Until now, subgoal labels have been given to students to learn passively. We report on a study to determine if giving learners subgoal labels is more or less effective than asking learners to generate subgoal labels within an introductory CS programming task. The answers are mixed and depend on other features of the instructional materials. We found that student performance gains did not replicate as expected in the introductory CS task for those who were given subgoal labels. Computer science may require different kinds of problem-solving or may generate different cognitive demands than mathematics or science.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education: computer science education, information systems education

General Terms
Measurement, Design, Experimentation.

Keywords
Subgoal labels, Cognitive Load, Contextual Transfer.

1. INTRODUCTION
As educators, we want to simplify the learning process to provide the maximum results. As researchers, we want find empirical evidence for what *exactly* it means to simplify the learning process. One proven method for enhancing learning is to reduce unnecessary cognitive load on the student while they are trying to learn to solve problems [22]. There are several ways to reduce cognitive load, including using worked examples [14].

Worked examples typically include a problem statement along with a step-by-step procedure for how to solve the problem. Worked examples are most effective when used in worked example-practice pairs [2]. In these pairs, students study a worked example solution and immediately practice by solving a similar problem.

Segmenting worked examples and including subgoal labels have also been shown to be effective in improving learning [2]. Segmenting includes separating portions of the worked example to isolate each step in the process [23]. Subgoal labels are names given to a set of steps in the solution process allowing the user to "chunk" the information to ease learning [10].

While these cognitive load reducing techniques have been empirically tested in math and science disciplines, we have been the first to test these with computer science learning [15]. Margulieux et al. [15] demonstrated learning benefits for subgoal labels with a drag-and-drop programming language. This paper reports on a study undertaken to empirically determine the effectiveness of worked examples and subgoal labels within introductory computer science using a more traditional textual language. Some of the findings confirm the results from other disciplines while some were unexpected.

Specifically, instructional material was created to teach introductory programming students about the process of using and writing a `while` loop to solve programming problems. There were three treatment conditions: (1) *no* subgoal labels provided, (2) subgoal labels *given*, and (3) subgoal labels *generated*, in which students were asked to generate their own labels for groups of solution statements. Within each treatment group, participants were randomly assigned to either an *isomorphic* or *contextual transfer* group. In the isomorphic transfer group, the problem to be solved in the worked example-practice problem pair was identical to the worked example in both procedural steps and cover story (i.e., context). The only thing changed was the actual values of the numbers to be calculated. In the contextual transfer group, the problem to be solved in the worked example-practice problem pair involved the same procedural steps but the cover story and numeric values changed. Participants' learning was measured with performance on novel problem solving tasks and a post-test. Problem solving tasks during the assessment were different from practice problems solved as part of the instructions.

The research questions to be addressed through this study were: How do students who generate their own subgoal labels perform compared to those who were given subgoal labels and those who learned without subgoal-oriented instructions? Does changing the context or "cover story" between the worked example and practice problem have an effect on learning?

2. BACKGROUND
In this section we review the current literature for cognitive load, worked examples, and subgoal labeling.

ICER '15, August 09-13, 2015, Omaha, NE, USA
ACM 978-1-4503-3630-7/15/08.
http://dx.doi.org/10.1145/2787622.2787744

2.1 Cognitive Load

Cognitive load can be defined as "the load imposed on an individual's working memory by a particular (learning) task" [28]. The cognitive load required to comprehend materials directly affects how much students learn, and affects their performance scores on assessments related to that task. If students have to keep too many things in working memory in order to understand a concept, learning suffers. As designers of instructional material, it is our responsibility to ensure that we do not overload the learner's working memory where possible when presenting new material. That is, we should help ensure that students' attention is directed at what's important for learning, rather than extraneous aspects of the material.

The central problem identified by Cognitive Load Theory (CLT) is that learning is impaired when the total amount of processing requirements exceeds the limited capacity of working memory [20]. Currently CLT [17, 24, 26] defines two different types of cognitive load on a student's working memory: intrinsic load and extraneous load.

Intrinsic load is a combination of the innate difficulty of the material being learned as well as the learner's characteristics [13]. Extraneous load is the load placed on working memory that does not contribute directly toward the learning of the material---for example, the resources consumed while understanding poorly written text or diagrams without sufficient clarity [13]. Working memory resources that are devoted to information that is relevant or germane to learning are referred to as 'germane resources' [25].

The intrinsic and extraneous loads can be controlled through instructional design. When designing instructional material care should be given to eliminate any possible extraneous load while attempting to minimize the intrinsic load. It is believed that worked examples, when carefully designed, can accomplish both of these goals [24].

2.2 Worked Examples

Worked examples are one type of instruction used to teach procedural process to students for problem solving activities. Worked examples give learners concrete examples of the procedure being used to solve a problem.

Eiriksdottir and Catrambone argue that learning primarily from worked examples does not inherently promote deep processing of concepts [12]. While it may result in better initial performance because examples are more easily mapped to problems, it is less likely result in the retention and transfer [12]. When studying examples, learners tend to focus on incidental features rather than the fundamental features because incidental features are easier to grasp and novices do not have the necessary domain knowledge to recognize fundamental features of examples [11]. For example, when studying physics worked examples, learners are more likely to remember that the example has a ramp than that the example uses Newton's second law [11]. A focus on incidental features leads to ineffective organization and storage of information that, in turn, leads to ineffective recall and transfer [6].

2.3 Subgoal Labels

To promote deeper processing of worked examples and, thus, improve retention and transfer, worked examples have been manipulated to promote subgoal learning. Subgoal learning refers to a strategy used predominantly in STEM fields that helps students deconstruct problem solving procedures into subgoals, functional parts of the overall procedure, to better recognize the fundamental components of the problem solving process [1]. Subgoals are the building blocks of procedural problem solving and they are inherent in all procedures except the most basic.

Subgoal labeling is a technique used to promote subgoal learning that has been used to help learners recognize the fundamental structure of the procedure being exemplified in worked examples [8–10]. Subgoal labels are function-based instructional explanations that describe the purpose of a subgoal to the learner. For example, in the problem in Figure 1 for the first two lines of code the subgoal label might read "Initialize Variables." This label provides information about the purpose of that subgoal and the function behind the steps within it. Studies [3, 4, 8–10, 15, 16] have consistently found that subgoal-oriented instructions improved problem solving performance across a variety of STEM domains, such as programming (e.g., [15]) and statistics (e.g., [10]).

Studies have found that giving subgoal labels in worked examples improves performance while solving novel problems without increasing the amount of time learners spend studying instructions or working on problems (e.g., [15]). Subgoal labels are believed to be effective because they visually group the steps of worked examples into subgoals and meaningfully label those groups [1]. This format highlights the structure of examples, helping students focus on structural features and more effectively organize information [2].

By helping learners organize information and focus on structural features of worked examples, subgoal labels are believed to reduce the extraneous cognitive load that can hinder learning but is inherent in worked examples [21]. Worked examples introduce extraneous cognitive load because they are necessarily specific to a context, and students must process the incidental information about the context even though it is not relevant to the underlying procedure [26]. Subgoal labels can reduce focus on these incidental features by highlighting the fundamental features of the procedure [21]. Subgoal labels further improve learning by reducing the intrinsic load by providing a mental organization (i.e., subgoals) for storing information.

Subgoal labels that are independent from a specific context have been the most effective type of subgoal labels in the past [7, 10]. Catrambone found that learners who were given labels that were abstract (e.g., Ω) and had sufficient prior knowledge performed better than those who were given labels that were context-specific (e.g., isolate x) on problem solving tasks done after a week long delay or in problems that required using the procedure differently than demonstrated in the examples [10]. Catrambone explained this exception by arguing that learners with sufficient prior knowledge were able to correctly explain to themselves the purpose of the subgoal and that by self-explaining the function of the subgoal--the self-explaining presumably due to the abstract label--was more effective than providing labels.

3. METHOD OF STUDY

3.1 Purpose

Participants in introductory programming classes were given instructional material designed to teach them to solve programming problems using `while` loops. This common introductory programming task requires only minimal prior programming knowledge (arithmetic operations and Boolean expressions) to complete at a basic level. The study was conducted before students had formally learned about `while` loops in their courses. Participants were recruited from 4 different introductory programming courses at a technical university in the southeast United States and the study was conducted over a two

week period. Because the courses teach different programming languages (see Table 1), pseudo-code was used in the task to make it independent from any one programming language.

Table 1. Classes Participating in Study

Programming Language	Majors
C++	Engineering
C#	Game Development
Java	Computer Science, Information Technology, Software Engineering, Non-Majors (mostly physics and math)

Pseudo-code is relatively easy for programmers to understand regardless of the programming languages that they know [27].The study was conducted in a closed lab setting with up to 30 computers in a single room. Students received an introduction to the study explaining that the material in the study was designed to help them learn how to write loops. Students were then given a URL to the first page of the study, which was housed in SurveyMonkey. Participants worked independently, but each session included between 15 and 30 people. The sessions typically lasted between 1 and 2 hours, depending on the rate at which participants completed the tasks.

3.2 Instructional Materials

To learn the procedure for using `while` loops to solve programming problems, participants were given three worked examples and three practice problems. The worked examples and practice problems were interleaved so that after studying the first worked example, participants solved the first practice problem before moving on to the second worked example. The worked examples came in three formats, which varied between participants. The first format was not subgoal oriented, meaning that steps of the examples did not provide any information about the underlying subgoals of the procedure. The second format grouped steps of the example by subgoal and provided meaningful subgoal labels for each group as is typical in subgoal label research (e.g.,[15]). The third format grouped steps of the example by subgoal and provided a spot for participants to write generated subgoal labels for each group. Each of the groups was numbered as "label 1," "label 2," etc., and groups that represented the same subgoal had the same number; therefore, groups that represented subgoal 1 were numbered as "label 1" regardless of where in the example they appeared (see Figure 1). Participants were told that each of the worked examples would have the same subgoals, and they were encouraged to update and improve upon their generated labels as they learned more.

Participant groups also received different practice problems to test how contextual transfer may affect learning. In the isomorphic transfer condition, the procedure and context used to solve the worked example and practice problem were exactly the same but the exact values in the problem changed. For example, if a worked example asked participants to find the average of quiz scores with values 70, 80, and 90, then the practice problem asked participants to find the average of quiz scores with values 75, 85, and 95. In the contextual transfer condition, the procedure used to solve the worked example and practice problem were the same except the context of the problem changed. For example, if a worked example asked participants to find the average of quiz scores, then the practice problem asked participants to find the average of money amounts. The contextual transfer was intended to be harder for participants to map concepts from the worked example to the practice problem. More difficult mapping can improve learning by

No labels	Given Labels (Passive)	Placeholder for Label (Constructive)

sum = 0
lcv = 1
 	
WHILE lcv <= 100 DO
 	
sum = sum + lcv
 	
lcv = lcv + 1
 	
ENDWHILE
	Initialize Variables
	sum = 0
	lcv = 1
	
	Determine Loop Condition
	WHILE lcv <= 100 DO
	
	Update Loop Var
	lcv = lcv + 1
	ENDWHILE
		Label 1: _____
		sum = 0
		lcv = 1
		Label 2: _____
		WHILE lcv <= 100 DO
		
		Label 3: _____
		lcv = lcv + 1
		ENDWHILE

Figure 1. Partial worked example formatted with no labels, given labels, or placeholders for generated labels.

reducing illusions of understanding caused by shallow processing thus inducing deeper processing of information [5, 12, 19]. However it can also increase cognitive load and potentially hinder learning [26].

After completing the instructions, participants completed novel programming tasks to measure their problem solving performance. We hypothesized that students who generated subgoal labels would learn better than those who were given the subgoal labels, and both groups would do better than those who had no subgoals at all. We also hypothesized that learners whose practice problems required contextual transfer would perform better than learners whose practice problems were the same context, unless the contextual transfer required too much cognitive load during the learning process.

3.3 Design

The experiment was a 3-by-2, between-subjects, factorial design: the format of worked examples (unlabeled, subgoal labels given, or subgoal labels generated) was crossed with the transfer distance between worked examples and practice problems (isomorphic or contextual transfer). The dependent variables were performance on the pre- and post-test, problem solving tasks, and time on task.

3.4 Participants

Participants were 66 students from a technical university in the Southeast United States (Table 2). Students were offered credit for completing a lab activity as compensation for participation. All students from these courses were allowed to participate, regardless of prior experience with programming or using while loops. To account for prior experience, participants were asked about their prior programming experience in high school (either regular or advanced placement courses) and college and whether they had experience using while loops. Other demographic information collected included gender, age, academic major, high school grade point average (GPA), college GPA, number of years in college, reported comfort with computer, expected difficulty of the programming task, and primary language. There were no statistical differences between the groups for demographic data, which is expected because participants were randomly assigned to treatment groups. Participants also took a multiple-choice pre-test to measure problem solving performance for using `while` loops. Average scores on the pre-test were low, 24% (1.2 out of 5 points), with 32% (21 out of 66) of participants earning no points.

Table 2. Participant Demographics

Age	Gender	GPA	Major
M = 21	89% male	*M* = 3.1/4	50% CS major

Many participants did not complete all tasks of the experiment. Participants received compensation regardless of the amount of time or effort that they devoted to the experiment, which might

have caused low motivation in some participants. Participants who did not attempt all tasks were excluded from analysis. Participants who answered more than two questions correctly out of the five on the pre-test were excluded from analysis because the instructions were designed for novices. To make the group size equal across conditions, an assumption of general linear model analysis, randomly chosen participants from some groups were excluded from analysis. Based on these exclusion criteria, we analyzed data from 66 of the 96 participants in the experiment.

3.5 Procedure

An outline of the entire study is given in Table 3. After granting consent (Step 1), the participants completed a demographic questionnaire (Step 2) and pre-test (Step 3). The pre-test was comprised of multiple choice questions about `while` loops from previous Advanced Placement Computer Science exams. Because the questions were multiple-choice, participants needed to only recognize correct answers rather than create correct answers.

When participants finished the demographic questionnaire and pre-test, they began the instructional period (Steps 4-6). The instructional period started with training. Participants who generated their own subgoal labels received training on how to create subgoal labels. The training included expository instructions about generating subgoal labels and an example of a subgoal labeled worked example similar to that in Figure 1. Then the training asked participants to complete activities to practice generating subgoal labels.

The first activity asked participants to apply the subgoal labels from the example to a new worked example. The second activity asked participants to generate their own subgoal labels for an order of operations math problem. After participants generated their own subgoal labels, they were given labels created by an instructional designer for comparison.

Participants who did not generate their own subgoal labels received training to complete verbal analogies. Verbal analogies (e.g., water : thirst :: food : hunger) were considered a comparable task to subgoal label training because they both require analyzing text to determine an underlying structure. Participants who were not asked to generate their own labels were not given subgoal label training because it might have prompted them to process the instructions more similarly than would be expected to participants who were asked to generate their own labels, which might confound the results. Like the subgoal label training, the analogy training included expository instructions, worked examples, and activities to carry out.

Following the training, the instructional period provided worked examples and practice problem pairs (Step 6) to help participants learn to use `while` loops to solve problems. The worked example format differed between subjects among three levels: unlabeled, subgoal labels given, and subgoal labels generated. Furthermore, the transfer distance between worked example and practice problem differed between subjects between two levels: isomorphic or contextual transfer. For a summary of the procedure during the instructional period, please refer to Table 3.

Having completed the instructional period, participants were then asked to complete a 10 item survey designed to measure cognitive load [18]. The placement of the cognitive load survey at this point is to ensure measurement of the actual learning process and not the assessment elements.

Table 3. Study Outline

Step	No Subgoal Labels		Subgoal Labels Given		Subgoal Labels Generated	
1	Consent					
2	Demographics					
3	Pre test					
4	Training Problem – summing					
5	Analogy Training & Activity				Subgoal Training & Activity	
Groups	**None-Isomorphic**	**None-Context Transfer**	**Given-Isomorphic**	**Given-Context Transfer**	**Generate-Isomorphic**	**Generate-Context Transfer**
6	Worked Example 1 (no subgoal labels)		Worked Example 1 (subgoal labels given)		Worked Example 1 (space to generate subgoal labels)	
	Problem 1 (no subgoal labels)	Problem 1A (no subgoal labels)	Problem 1 (subgoal labels given)	Problem 1A (subgoal labels given)	Problem 1 (space to generate subgoal labels)	Problem 1A (space to generate subgoal labels)
	Worked Example 2 (no subgoal labels)		Worked Example 2 (subgoal labels given)		Worked Example 2 (space to generate subgoal labels)	
	Problem 2 (no subgoal labels)	Problem 2A (no subgoal labels)	Problem 2 (subgoal labels given)	Problem 2A (subgoal labels given)	Problem 2 (space to generate subgoal labels)	Problem 2A (space to generate subgoal labels)
	Worked Example 3 (no subgoal labels)		Worked Example 3 (subgoal labels given)		Worked Example 3 (space to generate subgoal labels)	
	Problem 3 (no subgoal labels)	Problem 3A (no subgoal labels)	Problem 3 (subgoal labels given)	Problem 3A (subgoal labels given)	Problem 3 (space to generate subgoal labels)	Problem 3A (space to generate subgoal labels)
7	Cognitive Load Measurement					
8	Problem Solving Assessment (4 problems; 2 near transfer, 2 far transfer)					
9	Assessment Task 2					
10	Assessment Task 3					
11	Post Test					

Once participants completed the cognitive load survey, they started the assessment period (Steps 8-11). The assessment period included three types of tasks, but only the problem solving tasks (Step 8) will be discussed here because they are the only measure of novel problem solving performance. The problem solving tasks asked participants to use the problem-solving structure that they had learned during the worked example-practice problem pairs to solve four novel problems. Two of these problems required contextual transfer, meaning that they followed the same steps found in the instructions but in a different context, or cover story. The other two problems required both contextual and structural transfer. In these problems the context was new to the participants and the solution to the problem required a different structure than the problems found in the instructional material (e.g., the practice problem is summing values, the assessment is counting matching values). These tasks were intended to measure participants' problem solving performance as a 'far' transfer. After the assessment period, participants completed a post-test that had the same questions as the pre-test to measure their learning (Step 11).

Throughout the procedure, we recorded the time taken to complete each task. We also collected process data throughout the instructional period. We collected performance on the training activities and practice problems to ensure that participants were completing tasks. We also collected the labels that participants created.

We entered into the study with the following hypotheses:

H1. Participants who learn with subgoal labels (given or generated) will perform better on programming assessments and a post-test.

H1A. Those who generate their own subgoal labels and receive multiple variations of the problems (contextual transfer condition) will perform the best on the assessments, unless dealing with transfer overloads their mental resources.

H2. Participants who generate subgoal labels will perform better on problem solving tasks that require farther transfer. Those groups exposed to contextual transfer practice problems will perform better on transfer tasks than the isomorphic transfer groups.

H3. Participants who are given subgoal labels will complete the worked example-practice problem pairs in less time than others.

H3A. Those who generate subgoal labels and are exposed to contextual transfer practice problems will take the most time to complete the worked example-practice problem pairs.

H4. Participants with the deepest learning, those required to generate subgoal labels, should spend the least time on the programming assessments than other groups.

H4A. Participants with the most shallow learning, those with no subgoal labels and not exposed to contextual transfer problems, should spend the most time on the programming assessments.

4. ANALYSIS AND RESULTS

4.1 Accuracy

We scored participants' solutions for accuracy to generate a problem solving score. Participants earned one point for each correct line of code that they wrote. This scoring scheme allowed for more sensitivity than scoring solutions as wholly right or wrong. If participants wrote lines that were conceptually correct but contained typos or syntax errors (e.g., missing a parenthesis), they received points. We scored logic errors (having < rather an <=) as incorrect. We considered scoring for conceptual and logical accuracy more valuable than scoring for absolute syntactical accuracy because participants were still early in the learning process. Participants could earn a maximum score of 44.

The effect of the interventions on problem solving performance depended on the interaction of the worked example manipulation and transfer distance manipulation. We found no main effect of worked example format, F (2, 60) = 2.16, MSE = 123.5, p = .13, est. ω^2 = .07. In addition, we found no main effect of transfer distance, F (2, 60) = 0.04, MSE = 123.5, p = .83, est. ω^2 = .001. There was, however, a statistically significant interaction between worked example format and transfer distance, F (2, 60) = 6.5, MSE = 123.5, p = .003, est. ω^2 = .18, f = .31 (see Figure 2).

In this interaction the difference between the group that was given subgoal labels with isomorphic transfer (M = 12.1, SD = 13.5) and the group that was given subgoal labels with contextual transfer (M = 25.5, SD = 11.4) was statistically significant with a large effect size, t (20) = -2.51, p = .021, d = 1.07. Furthermore, the difference between the group that generated subgoal labels with isomorphic problems (M = 25.5, SD = 8.7) and the group that generated subgoal labels with contextual transfer (M = 17.5, SD = 11.5) was not statistically significant but had a medium effect size, t (20) = 1.86, p = .077, d = .78. These results mean that participants who were given subgoal labels performed better when they had contextual transfer, and participants who generated subgoal labels performed better with isomorphic problems.

We found three levels of performance, as can be seen in Figure 2. The best performing groups were those that were given subgoal labels with contextual transfer (M = 25.46) and generated subgoal labels with isomorphic problems (M = 25.55). The middle groups were those that received no labels with isomorphic problems (M = 18.09) and generated subgoal labels with contextual transfer (M = 17.46). The worst performing groups were those that received no labels with contextual transfer (M = 11.09) and were given subgoal labels with isomorphic problems (M = 12.09). The difference between the middle and best level of performance was not statistically significant but had a medium effect size, as shown by the t-test comparing groups that generated subgoal labels, t (20) = 1.86, p = .077, d = .78. Similarly, the difference between the middle and worst level of performance was not statistically significant but had a medium effect size, as shown by the t-test comparing groups that did not receive any subgoal labels, t (20) = 1.56, p = .13, d = .67. Given these effect sizes, we would expect these differences to be statistically different with a larger sample size.

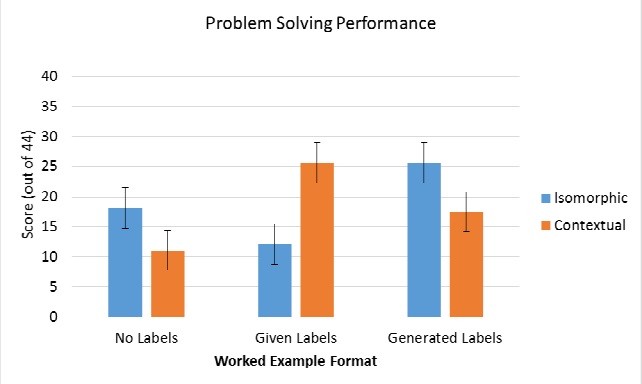

Figure 2. Problem solving performance graphed with worked example format on the x-axis, transfer distance as separate colors, and score on the y-axis.

Performance on the post-test was similar to that on the pre-test. Average scores on the post-test were low, 31% (1.5 out of 5 points). We found no statistical differences for main effect of worked example format, F (2, 60) = .39, MSE = 1.29, p = .68, est. ω^2 = .02, main effect of transfer distance, F (2, 60) = .83, MSE = 1.29, p = .37, est. ω^2 = .02, or interaction, F (2, 60) = 1.63, MSE = 1.29, p = .21, est. ω^2 = .06.

Some demographic characteristics correlated with performance on the problem solving tasks. Self-reported comfort with solving programming problems, collected on a Likert-type scale from "1 – Not at all comfortable" to "7 – Very comfortable," correlated positively with performance, r = .47, p < .001. Prior experience using while loops to solve programming problems, collected as a "yes" or "no" question, correlated positively with performance, r = .29, p = .018. Higher scores on these characteristics correlated with higher scores on performance. We found no differences among groups on these characteristics; thus, these correlations are not expected to confound the results.

4.2 Time Efficiency

4.2.1 Time on Worked Example-Practice Pairs

For time spent studying worked examples and solving practice problems, we found a main effect of worked example format, F (2, 60) = 6.55, MSE = 155.1, p = .003, est. ω2 = .18, f = .32. We also found a main effect of transfer distance, F (2, 60) = 6.24, MSE = 155.1, p = .015, est. ω2 = .09, f = .31. In addition, we found an interaction, F (2, 60) = 4.48, MSE = 155.1, p = .015, est. ω2 = .13, f = .26 (see Figure 3). Based on this pattern of results, the interaction is likely causing the main effect of transfer distance because there is little difference between transfer groups except when participants generated subgoal labels (see Figure 3).

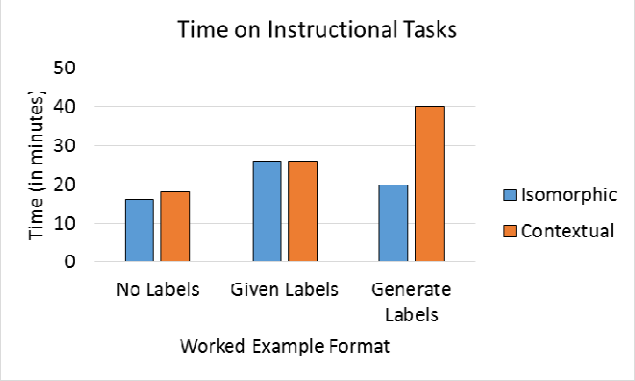

Figure 3. Time on instructional tasks graphed with worked example format on the x-axis, transfer distance as separate colors, and score on the y-axis.

4.2.2 Time on Programming Assessments

As in the results of the problem solving tasks, we found an interaction for time spent on the problem solving tasks, F (2, 60) = 3.97, MSE = 71.63, p = .024, est. ω2 = .12, f = .25 (see Figure 4). The main effect of worked example format was not statistically significant, F (2, 60) = .57, MSE = 71.63, p = .57, est. ω2 = .02, and we found no main effect of transfer distance, F (2, 60) = 1.34, MSE = 71.63, p = .25, est. ω2 = .02. This interaction is interesting because it almost exactly matches the pattern of problem solving performance so that more time on task maps to better performance. The exception is that the group that received no subgoal labels with isomorphic problems took the longest to complete the tasks but performed in the middle.

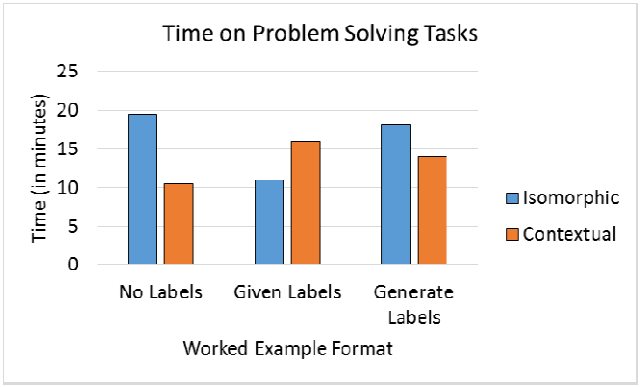

Figure 4. Time on problem solving tasks graphed with worked example format on the x-axis, transfer distance as separate colors, and score on the y-axis.

5. DISCUSSION

In this section we summarize our findings related to our original hypotheses. Table 4 contains a synopsis of all findings.

Table 4. Summary of Findings

Hypothesis	Finding
H1. Those with subgoal labels (given or generated) will perform better on programming assessments and a post-test.	Partially supported - Given-Isomorphic performed poorly
H1A. Those who generate subgoal labels and have contextual transfer in practice problems will perform the best on the assessments, unless transfer overloads their mental resources.	Generate-Context Transfer did better on the assessment, but not on the programming
H2. Participants who generate subgoal labels and those exposed to contextual transfer will perform better than other groups on problem solving tasks that require farther transfer.	Refuted
H3. Participants who are given subgoal labels will complete the worked example-practice problem pairs in less time than others.	Supported
H3A. Those who generate subgoal labels and have contextual transfer in practice problems will take the most time to complete the worked example-practice problem pairs.	Supported
H4. Participants required to generate subgoal labels, should spend the least time on the programming assessments.	Refuted
H4A. Participants with the most shallow learning, those with no subgoal labels and isomorphic practice problems should spend the most time on the programming assessments.	Supported - No subgoal labels and isomorphic transfer took the most time.

5.1 Accuracy

5.1.1 Assessments

Three groups performed the best on the assessments—combining the programming assessment and post test: those that were given subgoal labels with contextual transfer (Given-Context Transfer), and both groups that generated subgoal labels (Generate-Isomorphic and Generate-Context Transfer) (Figure 5).

Interestingly, the Generate-Context Transfer group did better on the post-test while the Generate-Isomorphic group performed better on the programming assessments. However the group that was given subgoal labels with no contextual transfer performed poorly on both the programming assessment and the post-test.

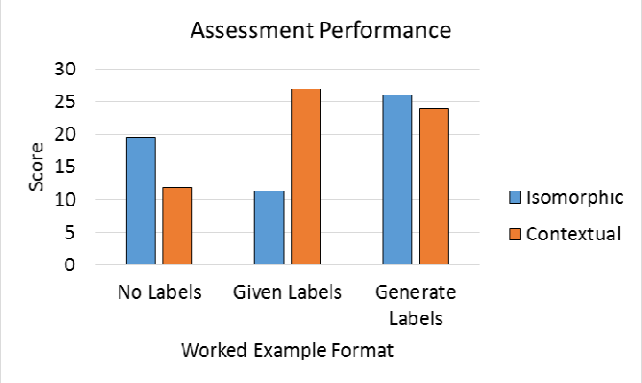

Figure 5. Assessment Performance by Treatment Groups

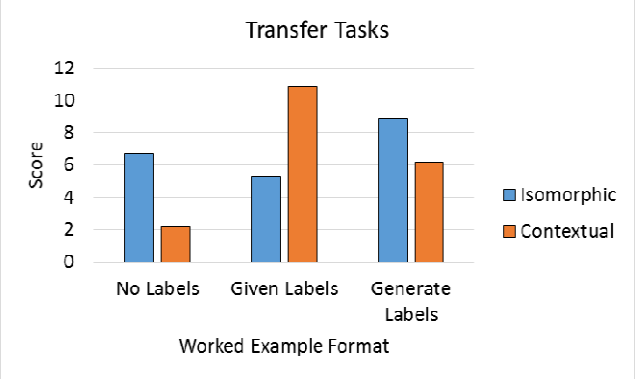

Figure 6. Transfer Task Performance

Thus we have partial support for H1. For the related hypothesis H1A, it was the case that the Generate-Context Transfer group performed statistically significantly better on the post-test assessment; they did not outperform the other groups on the programming assessment tasks. This may be because the generation of subgoal labels while also considering the contextual transfer overloaded the participants during the programming assessments when they were required to retrieve information from memory. However, performance on the post-test indicates that this group had the deepest learning when only considering conceptual recall and not problem solving issues.

When the commonalities between worked examples and practice problems were evident, as in the isomorphic transfer conditions, generating subgoal labels might have encouraged deep processing of information without overloading the participants. Similarly, when subgoal labels are given to participants, finding commonalities between contextually different examples and problems might have encouraged deep processing of information without overloading the participants. Participants who both generated subgoal labels and had contextual transfer did not perform as well as these groups. It is possible that both generating subgoal labels and finding commonalities between contextually different worked examples and practice problems was too cognitively demanding for many of the participants, which hindered performance.

5.1.2 Transfer Tasks

The best performing group on the transfer tasks (programming assessments 3 and 4) was the group that was given subgoal labels and contextually different practice problems (Given-Context Transfer) (see Figure 6). However the other two groups receiving contextual transfer practice problems did not perform particularly well on the transfer programming tasks and nothing was statistically different.

So we must refute H2. Those groups who were exposed to contextual transfer problems did not perform better than their isomorphic problem counterparts and this included the group that generated their own subgoal labels. However it should be noted that it was a contextual transfer group that did perform the best on the far transfer tasks – those that were *given* the subgoal labels.

5.2 Time

5.2.1 Worked Examples – Practice Problem Pairs

As expected, the group that took the most time on the instructional material of the worked examples and practice problem pairs was the group that had to generate their subgoal labels and contend with contextual transfer in the practice problems (Figure 3). This result was statistically significant and supports H3A. However we must refute H3. The given subgoal label groups did not complete the worked example-practice problem pairs in the least amount of time. In fact, it was the non-subgoal label groups who took the least time in completing the worked-example practice problem pairs. This may indicate that they were simply reading the material for shallow understanding. Notice also that the group with no subgoal labels and contextual transfer (None-Context Transfer) did take slightly longer than the None-Isomorphic group indicating that some time is likely spent translating the worked example solution into a new context.

5.2.2 Assessments

We have no support for H4 (see Figure 4). Indeed, the groups that spent the least amount of time on the programming assessments were the ones that received no subgoal labels with contextual transfer (None-Context Transfer) and the group that was given subgoal labels with no contextual transfer (Given-Isomorphic).

However, we have support for H4A. It was the group that did not receive any subgoal labels and no contextual transfer that took the most time on the programming assessment tasks.

5.3 Implications

Groups that generated subgoal labels performed overall better than those that did not have subgoal labels. The pattern of results for these groups is similar, though. In both cases, the condition that had isomorphic problems performed better than the condition that had contextual transfer, quite possibly because solving the isomorphic problems required less cognitive load. This pattern is reversed for groups that were given subgoal labels. It might be the case that learners who contend with contextual transfer problems need help identifying the analogous subgoals of the worked examples and practice problems. Participants who were given subgoal labels with contextual transfer might have been one of the highest performing groups because they received a framework of meaningful subgoal labels that guided their transfer between worked examples and practice problems. Though participants who generated subgoals labels received placeholders that indicated analogous subgoals between examples and problems, some of their generated labels were context-specific to the problem, which would not likely promote transfer to a contextually-different problem. In addition, if participants were unsure of the labels that they generated, they might rely less on them to guide future problem solving.

The most surprising result from this experiment was the group that was given subgoal labels and isomorphic problems was one of the worst performing groups. It could be that being given the

labels in addition to being able to more easily recognize commonalities between worked examples and practice problems led to superficial processing of information. Because participants could solve practice problems by using the worked example as an isomorphic guide and because the subgoal labels explained the function of programming steps, participants might have been overconfident about their understanding of the procedure and devoted less effort to learning.

We believe that there is an interesting interaction between the time spent during the instructional period and on the programming assessments that is related to performance. We now examine each group separately.

The None-Isomorphic group spent the least amount of time on the worked-example practice problem pairs which likely resulted in them spending the most amount of time on the programming assessment tasks. Their learning was most likely superficial learning which resulted in more thrashing when trying to solve the programming assessment tasks. And this group performed neither well nor poorly on the performance of the assessment tasks.

The None-Context Transfer group also spent the least amount of time on the worked example-practice problem pairs. However, they also spent the least amount of time statistically on the programming assessment tasks. This may be because these participants gave up and quit trying. For many of our participants it became obvious that if they felt they did not know the answer, they simply skipped attempting the task or put some form of "I don't know" for the result. While some did attempt the beginning of a solution-perhaps the first one or two lines of the solution, it was clear that they did not learn much overall.

The Given-Isomorphic group provides us the most puzzling results. We predicted that the Given-Isomorphic group would do well on the assessment tasks, based on previous research. However, this group performed the worst on the programming assessment tasks. Initially we thought it might be because this group was simply copying and pasting the results (the worked example problem and practice problem were on the same survey page). However, examination of their submissions show that the responses were not copied as the spacing is very different in their responses, some only entered the specific line related to the subgoal, and some wrote solutions in their "native" programming language rather than the pseudo-code. In addition, this group spent a fair amount of time during the instructional material period indicating that they were actually attempting to work through the solutions.

The Given-Context Transfer group is equally puzzling as they were among the best performing for the assessment tasks yet spent among the least amount of time on those assessments. These results are more in line with previous research – those that study worked examples can perform as well as those who solve problems in less time. It appears that this group internalized the most of the problem solving process allowing them to perform well on the assessments while not taking much time.

The Generate-Isomorphic group performed as expected on the assessment tasks – being among the best. However this group also took among the most time on the programming assessment. This may mean that they did not learn the material as deeply as the Generate-Context Transfer group or the Given-Context Transfer group.

The final group, the Generate-Context Transfer group behaved as expected related to previous research findings. They took the most amount of time while learning but also had among the best performance on the assessment tasks. It should be noted, however, that this group also had the most attrition amongst the groups (from an original number of 11 down to only 6 who completed the post-test). It may be that those who persisted until the end of the study are characteristically different than those who did not, so these results should be interpreted cautiously.

We collected and analyzed cognitive load component measurements using [18], however the differences were not statistically significant. No group reported significantly higher cognitive load, even though we know that generating subgoal labels requires more thought and mental effort than just reading and understanding given subgoal labels. Likewise, contextual transfer had no effect on the cognitive load component measures. This may be explained because all conditions had the same amount of intrinsic load, or because the measurement tool is not sensitive enough to capture the differences in this instance. This is definitely an area that needs further exploration.

6. CONCLUSION

The conclusion of these experiments is the colloquial expression, "There ain't no such thing as a free lunch." There are trade-offs in the design of learning opportunities. More time spent in learning does result in better performance later: Time on task matters for learning. If you spend less time on learning, students can still perform well on assessments. They will have to spend more time on the assessments to do as well.

Our findings continue to support the belief that subgoal labeling does improve learning. Generating those labels takes more time, and more time does result in more learning. However, being given labels may result in about the same amount of learning. In terms of efficiency (the most learning for the least amount of resources, including time), being given the subgoal labels may be the best option.

Having a context shift, from the example to the practice problem, has an interaction with subgoal labels in a way that is hard to explain. The best performance on the assessments comes from giving students the subgoal labels and requiring contextual transfer, or having students generate the subgoal labels but using only isomorphic transfer from example to practice.

The problem is that cognitive load in computer science is high due to the intrinsic nature of the material. Students have to keep in mind variables, their roles, their own process in problem-solving, and the process of the computer that they are attempting to model and control. While generating subgoal labels intuitively should lead to greater learning, there comes a point (e.g., if we add in contextual transfer) when the cognitive load of tracking everything makes learning difficult.

The intrinsic cognitive load of computer science is related to the languages we use (e.g., the fact that textual languages require *naming* of data and process, and we must remember and use those names) and the challenge of understanding and controlling a computational agent other than ourselves. That kind of problem does not occur frequently in science, mathematics, and engineering – but occurs from the very first classes in computer science. Because of this intrinsic load and the differences from other disciplines, we need to conduct replication studies. We cannot simply assume that findings from these other disciplines will predict learning in computer science.

The interventions for this study are strongly grounded in instructional design theory, and they were also applied in an authentic educational setting with an authentic educational task. Therefore, we expect that the internal and external validity of this

work is high. However, because this study is the first experiment to use this type of task and because the results were different than previous work with subgoal labels, research to replicate these results is needed to ensure the validity of this work.

7. ACKNOWLEDGMENTS

We would like to thank the students who participated in the study and their instructors who graciously gave us the time. We also thank the anonymous reviewers who supplied comments which improved this paper.

This work is funded in part by the National Science Foundation under grant 1138378. 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.

8. REFERENCES

[1] Atkinson, R.K., Catrambone, R., and Merrill, M.M., 2003. Aiding Transfer in Statistics: Examining the Use of Conceptually Oriented Equations and Elaborations During Subgoal Learning. *Journal of Educational Psychology*. 95, 4 (2003), 762.

[2] Atkinson, R.K., Derry, S., Renkl, A., and Wortham, D., 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research*. 70, 2 (2000), 181–214.

[3] Atkinson, R.K. 2002. Optimizing learning from examples using animated pedagogical agents. *Journal of Educational Psychology*. 94, 2 (2002), 416.

[4] Atkinson, R.K. and Derry, S.J. 2000. Computer-based examples designed to encourage optimal example processing: A study examining the impact of sequentially presented, subgoal-oriented worked examples. (2000).

[5] Bjork, R.A. 1994. Memory and metamemory considerations in the training of human beings. *Metacognition: Knowing about Knowing*. MIT Press.

[6] Bransford, J.D., Brown, A., and Cocking, R.R., 2000. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press.

[7] Catrambone, R. 1995. Aiding subgoal learning: Effects on transfer. *Journal of educational psychology*. 87, 1 (1995), 5.

[8] Catrambone, R. 1996. Generalizing solution procedures learned from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition; Journal of Experimental Psychology: Learning, Memory, and Cognition*. 22, 4 (1996), 1020.

[9] Catrambone, R. 1994. Improving examples to improve transfer to novel problems. *Memory & Cognition*. 22, 5 (1994), 606–615.

[10] 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.

[11] Chi, M.T., Bassok, M., Lewis, M.W., Reimann, P., and Glaser, R., 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science*. 13, 2 (1989), 145–182.

[12] Eiriksdottir, E. and Catrambone, R. 2011. Procedural instructions, principles, and examples how to structure instructions for procedural tasks to enhance performance, learning, and transfer. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. 53, 6 (2011), 749–770.

[13] Leppink, J, Paas, F., van der Vleuten, C., van Gog, T., and van Merriënboer, J., 2013. Development of an instrument for measuring different types of cognitive load. *Behavior research methods*. 45, 4 (2013), 1058–1072.

[14] Leppink, J., Paas, F., van Gog, T., van der Vleuten, C., and van Merriënboer, J., 2014. Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learning and Instruction*. 30, (2014), 32–42.

[15] Margulieux, L.E., Guzdial, M., and Catrambone, R., 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.

[16] Margulieux, L.E. and Catrambone, R. 2014. Improving problem solving performance in computer-based learning environments through subgoal labels. *Proceedings of the first ACM conference on Learning@ scale conference* (2014), 149–150.

[17] Van Merriënboer, J.J. and Sweller, J. 2005. Cognitive load theory and complex learning: Recent developments and future directions. *Educational psychology review*. 17, 2 (2005), 147–177.

[18] Morrison, B.B., Dorn, B., and Guzdial, M., 2014. Measuring cognitive load in introductory CS: adaptation of an instrument. *Proceedings of the tenth annual conference on International computing education research* (2014), 131–138.

[19] Palmiter, S. and Elkerton, J. 1993. Animated demonstrations for learning procedural computer-based tasks. *Human-Computer Interaction*. 8, 3 (1993), 193–216.

[20] Plass, J.L., Moreno, R., and Brünken, R., 2010. *Cognitive load theory*. Cambridge University Press.

[21] Renkl, A. and Atkinson, R.K. 2002. Learning from examples: Fostering self-explanations in computer-based learning environments. *Interactive learning environments*. 10, 2 (2002), 105–119.

[22] Renkl, A. and Atkinson, R.K. 2003. Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. *Educational psychologist*. 38, 1 (2003), 15–22.

[23] Spanjers, I.A., van Gog, T., van Merriënboer, J., 2012. Segmentation of worked examples: Effects on cognitive load and learning. *Applied Cognitive Psychology*. 26, 3 (2012), 352–358.

[24] Sweller, J., van Marriënboer, J., Paas, F., 1998. Cognitive architecture and instructional design. *Educational psychology review*. 10, 3 (1998), 251–296.

[25] Sweller, J., Ayres, P., and Kalyuga, S., 2011. *Cognitive load theory*. Springer.

[26] Sweller, J. 2010. Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational psychology review*. 22, 2 (2010), 123–138.

[27] Tew, A.E. and Guzdial, M., 2011. The FCS1: a language independent assessment of CS1 knowledge. *Proceedings of the 42nd ACM technical symposium on Computer science education* (2011), 111–116.

[28] van Gog, T. and Paas, F., 2012. Cognitive Load Measurement. *Encyclopedia of the Sciences of Learning*. Springer.