3-30-2021

Scaffolding Problem Solving with Learners' Own Self Explanations of Subgoals

Lauren Margulieux
Georgia State University

Richard Catrambone
Georgia Institute of Technology

Scaffolding Problem Solving with Learners' Own Self Explanations of Subgoals

Lauren E. Margulieux[1] and Richard Catrambone[2]

[1]Department of Learning Sciences, Georgia State University, Atlanta, GA 30302-3978

[2]School of Psychology, Georgia Institute of Technology, Atlanta, GA 30332-0170

Correspondence concerning this article should be addressed to Lauren Margulieux, Department of Learning Sciences, Georgia State University, Atlanta, GA 30302-3978. Email: lmargulieux@gsu.edu; Phone: 404-413-8064

Abstract

Procedural problem solving is an important skill in most technical domains, like programming, but many students reach problem solving impasses and flounder. In most formal learning environments, instructors help students to overcome problem solving impasses by scaffolding initial problem solving. Relying on this type of personalized interaction, however, limits the scale of formal instruction in technical domains, or it limits the efficacy of learning environments without it, like many scalable online learning environments. The present experimental study explored whether learners' self-explanations of worked examples could be used to provide personalized but non-adaptive scaffolding during initial problem solving to improve later performance. Participants who received their own self-explanations as scaffolding for practice problems performed better on a later problem-solving test than participants who did not receive scaffolding or who received expert's explanations as scaffolding. These instructional materials were not adaptive, making them easy to distribute at scale, but the use of the learner's own explanations as scaffolding made them effective.

Keywords: scaffolding; problem solving; distance education and telelearning; post-secondary education; programming and programming languages; teaching/learning strategies

Scaffolding Problem Solving with Learners' Own Self Explanations of Subgoals

Digital technology in face-to-face and online learning makes higher and lifelong education more accessible than ever before. Technology-supported face-to-face courses can include more students, and online learning provides access and flexibility that accommodates learners who live far from a university or work during the day. This accessibility is valuable as technical skills update more frequently and the number of people seeking technical skills increases. While many of these learning environments have great potential and enrollment (Smyth, Bossu, & Stagg, 2016), they have not been the drivers of education equality that providers had hoped (Downes, 2013; Littlejohn, Falconer, & Mcgill, 2008). Only a small percentage of students who already have effective learning and self-regulation strategies succeed without the personalized instructional support that is typical of smaller face-to-face courses (García Espinosa, Tenorio Sepúlveda, & Ramírez Montoya, 2015; Kizilcec, Pérez-Sanagustín, & Maldonado, 2017; Rohs & Ganz, 2015). Providing personalized instructional support while requiring fewer instructor resources, particularly to support learning technical skills and help student overcome problem solving impasses, is a large and active field of study (e.g., Aleven & Koedinger, 2002; Conati & VanLehn, 2000; VanLehn, 2011).

This study explores a method to support students while they complete computer programming problems independently. We expanded upon an instructional intervention that Margulieux and Catrambone (2019) found to help independent learners engage in beneficial learning strategies and self-regulation. Margulieux and Catrambone explored methods for promoting constructive learning through self-explanation of computer programming worked examples to help learners who did not have access to help from an instructor or peers to process new knowledge and connect it with prior knowledge. In the present study, we explored whether learners' self-explanations of worked examples can be used as effective scaffolds for initial problem solving to ultimately improve later problem-solving performance. If learners can self-explain examples and use those self-explanations to scaffold their initial problem solving attempts, then we can design learning environments to embrace constructivist learning principles, adapt for each learner's level of knowledge, and improve learning while requiring fewer instructional resources per student (Littlejohn et al., 2008).

1.1 Scaffolding Problem Solving

Scaffolding is a technique for bridging the gap between a learner’s current skill set and desired skill set by providing guidance to learners from knowledgeable sources, such as instructors and instructional designers (Kim & Hannafin, 2011). Scaffolding is a fundamental component of constructivism-based pedagogies (Pea, 2004). Constructivism theorizes that people build knowledge by constructing it for themselves through integrating new information into existing knowledge structures, rather than by being told what to know (Vygotsky, 1978; Wood, Bruner, & Ross, 1976). Learners who have limited prior knowledge, however, will need instructional support to build knowledge and reduce floundering (Hmelo-Silver, Duncan, & Chinn, 2007; Schmidt, Loyens, van Gog, & Paas, 2007). Much of current research on constructivist-based learning environments is devoted to exploring the appropriate types and amounts of guidance, including scaffolding, to support learning (Tobias & Duffy, 2009).

Scaffolding can be used in a variety of methods and in a variety of domains, ranging from giving a learner hints about the next step of solving a problem to leading a learner to recognize conceptual errors that have caused misconceptions (Pea, 2004). For this reason, scaffolding has many definitions, even when referring only to scaffolding in problem solving (Kim & Hannafin, 2011; Pea, 2004; Vygotsky, 1978; Wood, Bruner, & Ross, 1976). For the present study, we focus on conceptual scaffolding that adds more structure to problem solving than unscaffolded problem solving, in which the learner receives a problem without guidance about how to solve it (Hill & Hannafin, 2001). In particular, we used fixed scaffolding, which is pre-determined and does not adapt based on the learner's knowledge or performance (Azevedo, Moos, Greene, Winters, & Cromley, 2008; also called hard scaffolds; Saye & Brush, 2002).

Our scaffolding structured learners' initial problem-solving attempts by providing information about the pieces of the problem that they needed to solve to achieve a correct solution (see Figure 1). This scaffolding is reminiscent of the problem completion effect in learners who have more guidance during initial problem solving are better able to solve novel problems later (Sweller, 2010). The problem completion effect is based on cognitive load theory, which states that cognitive load (i.e., demand on learners' mental resources) has three sources:

1. information that is intrinsic to the task (e.g., the procedure to solve a problem),
2. extraneous information that is a byproduct of instruction (e.g., details that are specific to a problem but not necessary to conceptually understand the procedure),
3. and learning strategies to organize and retain germane information (e.g., developing schemas or connecting to prior knowledge).

The problem completion effect works by narrowing the problem-solving space, and, thus, reducing the amount of task information that the learner must process and leaving more cognitive resources for employing learning strategies (Sweller, 2010).

Scaffolding work typically explores scaffolding provided by humans, such as instructors or tutors, but recent studies have focused on providing scaffolding through technology (Delen et al., 2014; Kim & Hannafin, 2011). Humans can naturally provide adaptive scaffolding based on an evolving understanding of the learner's knowledge and progress (Azevedo, 2005). In adaptive scaffolding, input from learners helps scaffolders to adjust their instruction to match the learner's knowledge level (Yelland & Masters, 2007). This adaptive interaction between the scaffolder and learner makes creating technology that provides adaptive scaffolding difficult because it requires that the scaffolding system interpret the learner's knowledge such as through a cognitive model in an intelligent tutoring system (Aleven & Koedinger, 2002; Conati & VanLehn, 2000; VanLehn, 2011), and determine which pre-programmed option to give the user through a logic tree.

Incorporating adaptive scaffolding in large-scale problem-solving instruction is typically impractical, at least for the near future (Czerniewicz, Deacon, & Walji, 2018; Ossiannilsson, Williams, Camilleri, & Brown, 2015). The sophisticated system architecture and logic trees that enable adaptive scaffolding from technology are too technical and time-intensive to develop for varied uses and problem-solving procedures. Fixed scaffolding, however, is relatively easy to provide because it gives all learners the same scaffolds and, therefore, does not involve logic or developing probability-based models of students' knowledge. In some cases, fixed scaffolding can provide some of the same performance and self-regulation benefits as adaptive scaffolding (Azevedo et al., 2008; Delen et al., 2014). The issue with fixed scaffolding is that it is always the same, regardless of the learner's knowledge, and based on an expert's perspective of the correct way to solve a problem, regardless of the learner's perspective and prior experiences (Yelland & Masters, 2007).

In the present study, we explore a method for providing scaffolding to learners that is fixed, and, therefore, practical to distribute at scale, but that is also based on the learner's knowledge and prior experiences. Before attempting to solve problems, we asked learners to self-explain a worked example of the problem-solving process, meaning that they constructed explanations for why particular steps of the worked example were taken to reach the solution. We guided their self-explanations through a framework called subgoal learning, described in the next section. To scaffold learners' initial problem-solving attempts, we used a fixed structure (see Figure 1) to match learners' self-explanations to the problems. We compared this type of scaffolding to scaffolding with explanations by experts and unscaffolded problems.

We expected that scaffolding initial problem solving with learners' self-explanations would be more effective than scaffolding with experts' explanations because, though the scaffolding structure is fixed, the information in self-explained scaffolds would be more relevant to learners' understanding of the problem solution than an explanation by experts. This learner-centered scaffolding aligns with the theory of constructivism because the instructions reflect learners' own knowledge and explanations. Furthermore, using learners' explanations as scaffolding theoretically uses their cognitive resources more effectively in two ways. First, it reduces the amount of extra information that learners must process because the scaffolding includes learners' own thoughts rather than an instructor's explanation that must be interpreted by the learner. Second, it prompts learners to reflect on their explanations by applying them to new problems. The potential pitfalls of this approach stem from learners creating ineffective or inaccurate self-explanations; therefore, subgoal learning was used to support self-explanation.

1.2 Subgoal Learning through Self-Explanation

Subgoal learning is a framework used in procedural domains, like statistics and computer science, to help learners deconstruct problem solving procedures into subgoals (e.g., Catrambone, 1998; Margulieux & Catrambone, 2016). Subgoals are structural parts of a problem-solving procedure, in which the overall goal is to solve the problem. All procedures, except the most basic, can be deconstructed into subgoals. For instance, in Figure 2, the overall goal is to create an app in App Inventor. To do that, one would follow a series of steps. Each of those steps serves a sub-purpose, such as creating a component for the app or setting the properties of a component. Creating a component and setting properties are subgoals of the procedure to create an app. Highlighting the subgoals of a procedure helps learners to look past the contextual details of examples that cause extraneous cognitive load and focus on the structural components that are intrinsic to learning the problem solving process, improving learner retention and transfer to novel problems (Atkinson, Catrambone, & Merrill, 2003; Catrambone, 1998; Margulieux, Catrambone, & Guzdial, 2016).

Subgoal learning has primarily been promoted using subgoal labels (Catrambone, 1998; Margulieux et al., 2016; Margulieux & Catrambone, 2019). Subgoal labels are short, context-independent explanations that describe the purpose of a subgoal. In Figure 2, “Create Component” and “Set Properties” are subgoal labels that can be applied to any subgoal, regardless of context, that serves their function. Subgoal labels are typically used in worked examples to convey the purpose of a group of steps to the learner (e.g., Catrambone, 1998; Margulieux et al., 2016). Subgoal labeled worked examples have improved novel problem solving without increasing learners’ time spent studying instructions or solving problems (Margulieux & Catrambone, 2016; Margulieux et al., 2016). Prior research has suggested that subgoal labeled worked examples improve problem solving performance by helping learners to decontextualize examples, chunk information, and organize information (Atkinson, Derry, Renkl, & Wortham, 2000; Catrambone, 1995, 1996, 1998).

The drawback of subgoal labeled worked examples, however, is that they *provide* explanations to learners, making the subgoal learning process passive and dictated by experts rather than by learners. To align subgoal learning with the theory of constructivism and promote constructive learning, Margulieux and Catrambone (2019) supported subgoal learning through self-explanation of subgoals. Constructive learning, as defined by Chi (2009), requires that learners construct knowledge for themselves beyond the information provided for them via instruction. One method of constructive learning is self-explanation, in which learners use prior knowledge and logic to expand upon the instructions provided to them. For example, when learners self-explain the purpose behind the steps of a worked example, they are adding to the information provided in a worked example. Self-explanation can be more effective than instructor explanations because learners are integrating new knowledge with their prior knowledge (Wylie & Chi, 2014).

Margulieux and Catrambone (2019) compared passive subgoal learning (i.e., providing subgoal labels constructed by experts in worked examples) to constructive subgoal learning (i.e., asking learners to construct their own subgoal labels for worked examples). They provided different levels of support to learners who were constructing their own subgoal labels. Some participants received fixed *hints*, which were not adaptive to the learner, about the subgoals' purposes; some participants received fixed *feedback*, which was the expert-constructed subgoal labels, on the subgoal labels that they constructed; and some participants received worked examples already chunked into subgoals. The fixed supports that Margulieux and Catrambone used are compatible with the present study's paradigm of designing for problem solving instruction that can easily scale.

Out of Margulieux and Catrambone's (2019) constructive subgoal learning conditions, they found that participants who received worked examples that were already chunked into subgoals performed best when they received either hints (on subgoal labels to create) *or* feedback (on the subgoal labels that they created). Participants who received both hints *and* feedback performed worse than those that received only one form of support. Based on qualitative analysis of students' constructed labels, Margulieux and Catrambone argued that learners who received hints constructed high quality self-explanations, and that when this group received feedback in the form of expert-constructed labels, the learners misinterpreted the expert-constructed labels as the "correct" labels. Therefore, the learners disregarded the labels that they had constructed to adopt the expert-constructed labels, undoing all benefits of constructively learning the subgoals. Margulieux and Catrambone further argued that learners who did not receive hints constructed lower quality self-explanations and, thus, the feedback was beneficial to them because they used the expert labels to correct their own.

The present study expands upon this work by exploring whether the problem solving performance of the two best performing interventions from Margulieux and Catrambone (2019) -- 1) hints during subgoal label formation or 2) feedback (i.e., experts' explanations) on subgoal labels formed -- can be further improved by using learner-constructed subgoal labels to scaffold initial problem solving. By using learner-constructed subgoal labels to scaffold problem solving, the present study also provides more information about the quality and usefulness of self-explained subgoal labels.

1.3 Present Study

The present study explored whether subgoal labels constructed through self-explanation of worked examples could be used to scaffold initial problem solving to improve performance while solving novel problems and explaining problem solving procedures. The scaffolding of practice problems differed between participants. They either received unscaffolded practice problems or one of two types of scaffolded practice problems. The unscaffolded practice problems gave participants a problem to solve and a blank space to solve it. For the scaffolded practice problems, the blank space for solving the problem included scaffolds that sub-divided the problem-solving space in the form of subgoal labels that needed to be achieved (see Figure 1). The subgoal labels were either those constructed by the participants or those constructed by an instructional design expert and programming expert through a task analysis procedure (Catrambone, 2011). The labels constructed by the experts were expected to improve performance because they were similar to the guidance that an instructor might provide. Learner-constructed labels were also expected to improve performance, if participants constructed meaningful and accurate labels, because self-explanation helps learners mentally organize information (Wylie & Chi, 2014), and the labels would map that organization on to the problems. However, if participants struggled to self-explain the subgoals of the example, hastily made labels, or doubted whether their self-explanations were correct, then learner-constructed labels were not expected to provide guidance during problem solving. Therefore, the effect that learner-constructed labels had on guiding initial problem solving was expected to provide information about the efficacy of learner-constructed labels.

The problem-solving domain for the study was computer programming. Programming was considered an appropriate domain because it requires procedural problem solving, and programming instruction typically relies upon worked examples and practice problems. Furthermore, programming was a suitable domain for this experiment because previous work has found that programming performance can be improved by self-explanation of procedural structure (Pirolli & Recker, 1994; Soloway, 1986) and subgoal learning (Margulieux et al., 2016; Margulieux & Catrambone, 2016).

Participants were required to be programming novices to minimize confounds due to learners’ prior knowledge. Because participants were novices, they solved programming problems using a drag-and-drop programming language rather than a text-based language. Drag-and-drop languages provide users with menus of code blocks that include all the syntax and structure that novice programmers find difficult to learn in text-based languages (Hundhausen, Farley, & Brown, 2009). Then novices need to only piece together the code blocks like puzzle pieces and fill in situational details, like variable names or the number of times that for loop should repeat (see Figure 3). Drag-and-drop programming languages are popular for novices because they have low barriers to adoption (Grover & Pea, 2013). Entire open curriculums are being developed for Scratch (e.g., Maloney, Peppler, Kafai, Resnick, & Rusk, 2008), Android App Inventor (Grover & Pea, 2013), and many others. We used Android App Inventor, a language that can be used to create applications (apps) for Android devices, because people of all ages and genders are equally interested in app development (Grover & Pea, 2013). Participants used App Inventor to create a Music Maker app that has images (e.g., drum or cymbal) that play sounds when pressed. Many introductory programming courses use text-based languages rather than drag-and-drop languages, but recent research (Morrison, Margulieux, & Guzdial, 2015; Morrison, Decker, & Margulieux, 2016) suggests that the same self-explanation of subgoal learning intervention is effective for text-based languages too; thus, the effects are expected to generalize to those settings as well.

2. Method

2.1 Design

The experiment was a between-subjects design, meaning that all participants were randomly assigned, while controlling for number of participants per condition, to one of the six conditions. The first variable, subgoal learning method, had two levels based on findings from Margulieux and Catrambone (2019): with hints and without feedback or with feedback and without hints. For the rest of the paper, we will refer to these groups at the hints group and the feedback group. The second variable, practice problem scaffolding, had three levels: unscaffolded, scaffolded with learner-constructed labels, and scaffolded with expert-constructed labels. The main dependent measures were performance on problem solving and explanation tasks. Time on task was measured while participants engaged with the worked example (including subgoal label construction and re-creating the app), feedback or re-reading, practice problems, problem solving tasks, and explanation tasks. Besides the manipulated variables, the following variables were included in analysis as possible predictors of performance: learner characteristics, working memory capacity, pre-test score, cognitive load, post-test score, and perception of understanding. In addition, the subgoal labels that participants construct were analyzed qualitatively and considered as a possible predictor of performance.

2.2 Participants

The six conditions each had 20 participants (*N* = 120). Participants were undergraduate students at a southeastern US, mid-sized institute and recruited through a subject pool. Participants could not have prior experience using App Inventor nor have taken more than one computer science course in high school or college. People who had more prior experience were disqualified because the instructional materials were designed for novices. To ensure that participants did not have prior knowledge of the task, they completed a 5-item, multiple-choice pre-test that asked them about basic App Inventor functions. One answer choice for each item was “I don’t know,” and participants were encouraged to select this option unless they knew the answer. Having this option reduces the chance that participants appeared more knowledgeable than they were by guessing.

Learner characteristics that have been identified as possible predictors of programming performance were collected (i.e., age, gender, high school GPA, college GPA, year in school, computer science experience, comfort with computers, and expected difficulty of learning App Inventor; Rountree, Rountree, Robins, & Hannah, 2004). These characteristics did not correlate with performance (see Table 1), meaning that learner characteristics did not affect whether the intervention was effective or not.

In addition to these learner characteristics, working memory capacity was measured because it has been linked to successful self-explanation (Wylie & Chi, 2014). Working memory capacity was measured with the Shapebuilder task (Atkins et al., 2014) because it is similar in kind to programming with drag-and-drop programming languages. The Shapebuilder task shows participants a screen with four sets of four shapes, each set in a different color (i.e., 16 options total), and a 4x4 grid. The system then shows the participant a sequence of colored shapes on the grid (e.g., red circle in space 3 then blue circle in space 6) and the participant recreates the sequence from memory by dragging and dropping the shapes onto the grid, matching the order, shape, color, and location for each item. The task adds items to the sequence to make it progressively harder until it is confident that the participant cannot score higher. The task gives partial points for partially correct answers, like a correct color but incorrect shape. The Shapebuilder task matched the block-based programming task well because both involved dragging and dropping shapes in a correct sequence. Performance on the Shapebuilder task was not correlated with problem solving performance (see Table 1).

2.3 Experimental materials

This section describes the materials that participants received throughout the experiment. The materials were the same as those used in Margulieux and Catrambone (2019). They are listed in the order that participants received them.

2.3.1 Subgoal label training. All participants received the same subgoal label training because all participates creating their own subgoal labels, regardless of which manipulation they received. The training explained what subgoals and subgoal labels are, gave a worked example of constructing subgoal labels, and asked participants to construct subgoal labels for a simple math, order of operations, example problem. Order of operations was chosen because it is a task that all participants should know how to do and find easy. The last step of the training gave participants expert-constructed labels for the order of operations problem and asked them to compare their labels to the expert-constructed labels. The instructions specifically stated, “If your labels are different than the expert’s, that doesn’t mean that your labels are wrong.”

2.3.2 App Inventor instructions. The instruction for teaching App Inventor had a worked example that demonstrated the steps taken to create an app. The app played sounds when images of instruments were touched or the device was tilted or shaken. For instance, when a drum image or cymbal image was touched, a drum sound or cymbal sound, respectively, would play. Because creating the app was a long process, taking about 30 to 35 minutes, participants learned to create only one app and, therefore, received only one worked example. The worked example was visually grouped into subgoals (see Figure 4), and participants were asked to construct subgoal labels for each of the subgoals in the worked example. The worked example had five unique subgoals that were repeated at least four times throughout the worked example: create components, set properties, handle events, set output, and set conditions. Margulieux and

Catrambone (2019) identified these subgoals using the Task Analysis by Problem Solving (TAPS) procedure (Catrambone, 2011).

Each subgoal that served the same function was numbered with the same number (e.g., each group of steps that dealt with the ”set output” subgoal were numbered with “Function 4”) so that participants knew which subgoals served the same functions. This guidance was included to tell all participants which subgoals were functionally the same to help them to see past superficial differences in individual problem steps and focus on the function of groups of steps. Participants were asked to construct their subgoals after they had read through the entire worked example so that they saw each instance of the subgoals before constructing the labels.

Half of the participants received hints while constructing subgoal labels for the worked example. The hints highlighted similarities among all subgoals that served the same function and appeared once next to the first instance of each subgoal. See Figure 5 for the full list of hints and their corresponding subgoals. The hints were written to not be function-focused so that participants could not copy the hints to be their constructed subgoal labels. The other half of participants received feedback on the subgoal labels that they had constructed after they finished with the worked example. For feedback, participants received another copy of the worked example with expert-constructed subgoal labels instead of subgoal label placeholders (see Figure 2).

2.3.3 Practice problems. Participants solved four practice problems to ensure that they understood the problem-solving procedure before they attempted the five novel problems that made up the problem-solving assessment. Two of the practice problems required isomorphic transfer from the worked example, meaning that they differed from the worked example only in surface features. Both the procedural steps and the context of the problem matched the worked example. For instance, the worked example demonstrated how to create a drum ImageSprite that plays a drum sound when touched. Similarly, an isomorphic practice problem asked participants to create a cymbal ImageSprite that plays a cymbal sound when touched. The remaining two practice problems required contextual transfer from the worked example, meaning they differed in contextual features though the procedural steps still matched the worked example. For instance, one contextual transfer practice problem asked participants to create a button that changed colors when touched, which is similar to creating a drum ImageSprite that plays a sound when touched. Both require touching an area of the screen as an input from the user to prompt a noticeable, simple output from the device.

Participants were assigned to receive one of three versions of the practice problems (see Figure 1). One version was unscaffolded, meaning that it gave the problem statement and a blank space to solve it (left side of Figure 1). Another version was scaffolded with references to the participant's constructed labels. After the problem statement, the problem-solving space listed each of the subgoals that needed to be achieved to complete the problem with space between each subgoal label for the participant's work (middle of Figure 1). The last version was scaffolded with expert-constructed subgoal labels. After the problem statement, the problem-solving space listed the expert-constructed subgoal labels for each subgoal that needed to be achieved (right side of Figure 1).

2.3.4 Learning and manipulation checks. For the learning check, participants took a post-test that had the same items as the pre-test to ensure that they had learned the fundamentals of App Inventor. For the manipulation check, participants indicated which learning strategies they used during the instructional period from the following list: read the instructions, applied knowledge to complete practice problems, self-explained parts of the instructions, wrote down my own explanations of the instructions, compared my explanations to provided explanations, other (please explain). Along with these checks, participants rated their understanding of the app creation process and their comfort with creating simple apps in App Inventor. Participants also completed a cognitive load measure specifically adapted for programming instruction (Morrison, Dorn, & Guzdial, 2014). The measure asked three questions for intrinsic cognitive load (i.e., load that is necessary to learn the procedure), three questions for extraneous cognitive load (i.e., load that is necessary to complete the learning task but not to learn the procedure, such as contextual details of a worked example), and four questions for germane cognitive load (i.e., load that is necessary for using learning strategies).

2.3.5 Problem solving assessment. To measure problem solving performance, participants were given an assessment that asked them to solve five novel problems. The problems asked them to modify or create components of an app. Of the five problems, two required contextual transfer from the worked example, meaning that they differed from the worked example in context (e.g., creating a ball instead of a drum) but the procedural steps taken to solve the problem were the same. The other three problems required procedural transfer, meaning that they differed from the worked example in the exact steps taken to solve the problem, but the abstracted procedure was the same. For instance, the worked example listed steps to program a drum sound to play when an ImageSprite was touched, and a procedural transfer problem asked participants to program a label to display text when an ImageSprite was touched. The abstract procedure to create both functions was the same even though the individual steps were different.

2.3.6 Explanation tasks. Participants also completed explanation tasks that measured whether they could match steps of problem solutions to their functions. After participants completed the entire problem-solving assessment, they received, for each problem, the solution and a list of functions completed in the problem, which were the subgoal labels that the expert had constructed. Then they were asked to match solution's steps to the subgoal label that explained the function of that step. The instructions for the task clarified that multiple steps could match a single function, but multiple functions could not match a single step. This task used the solutions to the problem-solving assessment so that both assessments had the same context, reducing the contextual information unrelated to the problem-solving procedure that participants needed to process.

2.4 Procedure

Sessions took between 80 and 110 minutes, depending on how quickly participants complete each of the tasks. Before starting the instructional period, participants completed the demographic questionnaire, working memory measure, and pre-test, which took 10 to 15 minutes.

The instructional period took 40 to 55 minutes and contained all instructional manipulations. It started with a video that showed a person interacting with and explaining the various features of the App Inventor interface to introduce the interface to participants. This video was the same for all participants and did not include information about the procedure being taught. Palmiter, Elkerton, and Baggett (1991) found that videos help participants to intuitively learn to use direct manipulation interfaces, such as App Inventor. After the video, participants completed the subgoal label training and received the App Inventor instructions with the worked example of creating an app. Participants were assigned to receive one of two formats of the worked example: either with hints or without hints (see Figure 4). Participants also used the worked example to re-create the app in App Inventor and, thus, actively engaged in the instruction.

When participants finished re-creating the app and constructing their subgoal labels, participants who did not receive hints were given feedback. For feedback, participants received a copy of the worked example that had expert-constructed subgoal labels instead of subgoal label placeholders (see Figure 2). Participants were asked to compare the subgoal labels that they had constructed to those constructed by an expert. The other group, participants who received hints, did not receive feedback because in prior research (Margulieux & Catrambone, 2019), participants who received hints performed better when they did not receive feedback. To make time on task comparable between the groups, participants who received hints were asked to re-read the worked example. Therefore, differences between the groups cannot be attributed to differences in time on task.

Before receiving the assessment tasks, participants solved practice problems in the App Inventor interface to check their understanding. During this time, participants could access all the instructions that they had received. For the feedback condition, these instructions included the worked example labeled with expert-constructed subgoal labels. Participants solved four practice problems that typically took 10 to 15 minutes in total.

When participants finished the practice problems, all the instructional materials that they had been given was collected, but they still had access to App Inventor. Participants were informed at the beginning of the session and before receiving the practice problems that they would not have access to the instructions during the assessment but that they would be able to use App Inventor. Participants then received the cognitive load questionnaire. The questionnaire was given after the instructional period rather than after the assessment period to ensure that we were measuring cognitive load while using the instructions. Afterward, participants completed the learning and manipulation checks.

For the first assessment, participants received 25 minutes to complete the problem-solving assessment. Similar to an exam, participants did not have unlimited time to work on the tasks, but the tasks were piloted to ensure that participants had adequate time to work on problems. Participants could use App Inventor to help them work through their solutions, but ultimately they were asked to write down the steps that they took to complete the tasks. Writing down the steps allowed the scorers to see the steps that participants took toward the solution rather than only the final product. In addition, some parts of the solutions depended on correct completion of earlier parts of the solution (e.g., participants could not program a sound to play when the phone was shaken if they did not correctly create an accelerometer). Writing down the steps allowed participants to receive credit for these parts of the solution, even if they could not implement them in the interface. Before leaving the sessions, the participants took three to four minutes to complete the explanation tasks.

3. Results and Discussion

Before the instructional period, most participants (84%) scored zero points on the pre-test, and the remaining participants scored one point out of the possible five. After the instructional period, most participants (88%) correctly answered all five questions on the post-test, which had the same items as the pre-test, and the remaining participants scored four out of five points, suggesting that all participants paid attention to the instructions. No participants were removed from analysis based on pre-test or post-test score.

3.1 Learner-constructed labels effectively scaffold problem solving performance

Participants' solutions on the problem-solving assessment were scored for correct steps taken to solve the problems. Because the solutions required many steps, scoring solutions for correct steps rather than correct outputs provided more sensitivity, like grading for partial credit. Steps were at the grain-level of those in Figure 2. The maximum score was 25. Differences among condition were statistically tested with a two-way ANOVA and Bonferroni post hoc analyses when necessary.

Scaffolding of practice problems affected performance, $F(2, 114) = 9.81$, MSE = 21.5, $p < .001$, partial $\eta^2 = .15$, $f = .40$ (see Figure 6). Subgoal learning method did not affect performance, $F(2, 114) = 0.19$, MSE = 21.5, $p = .67$, partial $\eta^2 = .002$, nor was there an interaction, $F(2, 114) = 1.04$, MSE = 21.5, $p = .35$, partial $\eta^2 = .02$. To explore the effect of scaffolding across the three levels (i.e., no scaffolding, scaffolding with learner-constructed labels, or scaffolding with expert-constructed labels), a Bonferroni post hoc analysis was used. Participants whose practice problems were scaffolded with their own labels ($M = 21.6$, $SD = 3.86$) performed statistically better on later problem solving than those whose practice problems were scaffolded with labels made by experts ($M = 17.1$, $SD = 4.95$; Mean Difference = 4.55, $p < .001$) or had no scaffolding (M = 18.8, SD = 3.90; Mean Difference = 2.80, $p = .024$). To the researchers' surprise, participants whose practice problems included expert-constructed labels for scaffolding did not perform better than those who received no scaffolding, Mean Difference = 1.75, $p = .28$. The results show that scaffolding initial problem solving with learners' own explanations of the subgoals of the procedure resulted in better problem solving than no scaffolding or scaffolding with an expert's explanation of the subgoals of the procedure. Moreover, the difference had a large effect size (i.e., partial $\eta^2 = .15$, $f = .40$), suggesting that adding learner-constructed labels as scaffolding to practice problems can greatly improve problem solving performance over practice problems without scaffolding or even those with some types of expert-constructed scaffolding.

3.2 Analysis of subgoal label quality

To better understand why participants who constructed their own labels performed better when scaffolded with their labels, the learner-constructed labels were analyzed to determine their quality and whether certain characteristics of the labels corresponded with better performance. Each label that participants constructed was analyzed as one unit, meaning each word within a label was not analyzed individually. In addition, after finding through initial review that participants tended to make the same type of labels for each of the five subgoals, we decided to characterize each participant based on all their labels collectively. The coding scheme was determined *a priori* based on Margulieux and Catrambone's (2019) scheme, which characterized labels as context-specific, context-independent, or incorrect (see Table 2). Context-specific labels included contextual details about the specific instantiation of the subgoal and, therefore, was applicable to only that one instance. For example, the learner-constructed label "create the drum image" was characterized as context-specific because could be applied only to the steps that created the drum Image Sprite.

In contrast, context-independent labels did not include contextualized details about the specific instantiation of the subgoal and, therefore, could be applied to any instance of the subgoal. For example, the learner-constructed label "add components" was characterized as context-independent because it could be applied to any component that is added to the app, be it a drum, cymbal, or accelerometer sensor. Incorrect subgoal labels were either execution-based instead of function-based, such as "click on Image Sprite," or did not correctly describe the function, such as “start drum.” In all cases, participants who made incorrect labels had no context-specific or -independent labels.

We argue that context-independent labels promote learning better than context-specific labels because they suggest that learners have a more conceptual understanding of the procedure that can be seamlessly applied to solving new problems. Constructing context-independent labels is especially important when the labels are used to scaffold new problems. For example, if a novel problem was scaffolded with “add components,” the learner would be prompted to add the components for that problem. A scaffold that said “create the drum image” would require that the learner make an analogy between the drum image in the original problem, if they could remember it, and the components in the new problem.

To establish reliability in the ratings, two raters applied the coding scheme to labels from 20% of the participants. Then, interrater reliability was tested with intra-class correlation coefficient of agreement, ICC(A), because the coding scheme was based on nominal categories and absolute agreement would be the only valid standard for reliability. Initial interrater reliability well above threshold, ICC(A) = .96; therefore, the remaining labels were scored by a single rater. In all but three cases, each participant’s labels were either all context-specific, all context-independent, or all incorrect. In the other three cases, participants were classified based on which type of subgoal label they made most often.

Almost all participants constructed correct, function-based subgoal labels, which suggests that the subgoal label training provided adequate instruction for this activity. Only 4% of participants who received hints and 7% of participants who did not receive hints made incorrect labels. The major difference between the groups was that more participants who received hints made context-independent labels (75%) than those who did not receive hints (35%). The hints, therefore, helped learners to separate the context of the worked example from the functions demonstrated in it and make labels that transcended a single problem.

Though participants who received hints made context-independent labels more often, they did not perform better than those who did not receive hints in general, $F(2, 114) = 0.19$, MSE = 21.5, $p = .67$, partial $\eta^2 = .002$. If we look specifically at the groups that were scaffolded with their own labels, however, participants who received hints performed statistically better than those who did not in a 1-tailed *t*-test, $t(38) = 1.77$, $p = .032$, $d = .56$. Therefore, hints helped participants more than feedback when participants were scaffolded with their own subgoal labels.

Contrary to our expectations, subgoal label quality was not a predictor of problem-solving performance. Because subgoal label quality was not an independent variable, linear regression was used instead of ANOVA with problem solving performance as the dependent measure and format of practice problems, method of subgoal learning, and learner-constructed subgoal label quality as the predictors. The regression was stepwise with the predictors being entered into the equation in the order listed because previously reported ANOVAs and *t*-tests suggest that format of practice problems and method of subgoal learning are predictors of performance. The regression determined that subgoal label quality did not further predict problem solving performance, $\beta = -.08$, $t(3, 118) = -.77$, $p = .44$.

We expected that participants who constructed context-independent labels would perform better than those who constructed context-dependent labels on later problem solving because context-independent labels indicate a context-independent understanding of the procedure. However, we did not find evidence that constructing context-independent labels improved performance. Therefore, all participants who received their own self-explanations as scaffolding, regardless of which type support they received while self-explaining or the quality of self-explanations, performed better than those who received no scaffolding or expert's explanations as scaffolding. This finding supports this application of constructivism because it suggests that when learners' construct their own explanations with sufficient support, those explanations are better at supporting students than an expert's explanation. In this case, sufficient support included fixed guidance or feedback developed through instructional design, but it did not include personalized instruction at any point.

It is possible, however, that participants' labels were not representative of their final knowledge state. Participants constructed labels before receiving feedback, if they received feedback, and before working on practice problems. For the learners who initially constructed context-dependent labels and received feedback in the form of context-independent expert labels, perhaps that feedback helped them to recognize similarities among different instances of subgoals and decontextualize their understanding of the subgoals. In addition, working through the practice problems might have helped participants to decontextualize their subgoals. We cannot provide evidence for these possibilities though because participants did not externally update their labels at the end of the instructional period; therefore, we do not know how their understanding evolved over time. What we do know is that participants who started with lower quality explanations benefited more from scaffolding with their own explanations than from expert explanations.

3.3 No differences in time on task

The time taken to complete each part of the experimental procedure was recorded. No significant differences among conditions were found for time on task. On average, participants spent 34.1 minutes (*SD* = 5.99) completing the subgoal training, studying the worked example, and constructing subgoal labels. There was no main effect of subgoal learning method, *F*(2, 114) = 0.17, *MSE* = 36.2, *p* = .68. Effects of the other independent variable, format of practice problem, was not analyzed because participants had not had practice problems at this point. To solve the practice problems, participants took an average of 9.57 minutes (*SD* = 3.21). There was no main effect of practice problem format during this period, *F*(2, 114) = 1.12, *MSE* = 10.4, *p* = .33. This result means that the different types of scaffolding did not affect the amount of time participants took to solve the practice problems. There was also no main effect of subgoal learning method, *F*(2, 114) = 0.37, *MSE* = 10.4, *p* = .54, and no interaction, *F*(2, 114) = 0.65, *MSE* = 10.4, *p* = .52. During the problem-solving assessment, participants took an average of 23.2 minutes (*SD* = 2.58). There was no main effect of practice problem format, *F*(2, 114) = 0.56, *MSE* = 6.78, *p* = .60, no main effect of subgoal learning method, *F*(2, 114) = 0.34, *MSE* = 6.78, *p* = .56, and no interaction, *F*(2, 114) = 0.87, *MSE* = 6.78, *p* = .42. Therefore, the participants who received their own labels as scaffolding did not take longer to solve the practice problems or complete the problem-solving assessment than participants in other conditions, but they performed better on the problem-solving assessment.

3.4 No differences in other metrics

For the explanation assessment with a maximum possible score of 20, the mean score was 16.5 (*SD* = 3.67). There was no main effect of practice problem format, *F*(2, 114) = 0.18, *MSE* = 16.7, *p* = .84, subgoal learning method, *F*(2, 114) = 0.76, *MSE* = 16.7, *p* = .39, and no interaction, *F*(2, 114) = 1.35, *MSE* = 16.7, *p* = .26. These results suggest that all participants were equally prepared to complete the explanation task, regardless of whether they received hints during the worked example or received feedback after the worked example. The average score was close to the maximum score; therefore, most participants performed well.

No differences were found among conditions for cognitive load. Cognitive load was measured by self-report after the instructional period, including solving practice problems. The mean score was 39.7 (*SD* = 12.3) out of 100, with no main effect of practice problem format, *F*(2, 114) = 0.85, *MSE* = 155.4, *p* = .43, no main effect of subgoal learning method, *F*(2, 114) = 0.37, *MSE* = 155.4, *p* = .54, and no interaction, *F*(2, 114) = 0.20, *MSE* = 155.4, *p* = .82. In addition, no differences were found within each of the three types of cognitive load: intrinsic, extraneous, and germane (see Table 3). These results suggest that participants in different conditions experiences the same perceived cognitive load.

After rating their cognitive load, participants also rated how well they understood the instructions from “1 – Not well at all” to “7 – Very well.” Participants felt that they understood well (*M* = 6.00, SD = 1.0). There was no main effect of practice problem format, *F*(2, 114) = 1.99, *MSE* = 0.96, *p* = .14, subgoal learning method, *F*(2, 114) = 0.54, *MSE* = 0.96, *p* = .47, and no interaction, *F*(2, 114) = 0.61, *MSE* = 0.96, *p* = .55. Furthermore, participants rated how comfortable they were solving novel problems from “1 – Not comfortable at all” to “7 – Very comfortable.” Participants were comfortable solving novel problems (*M* = 5.8, SD = 1.0). There was no main effect of practice problem format, *F*(2, 114) = 2.44, *MSE* = 1.08, *p* = .09, subgoal learning method, *F*(2, 114) = 0.21, *MSE* = 1.08, *p* = .65, and no interaction, *F*(2, 114) = 0.08, *MSE* = 1.08, *p* = .93. These results suggest that participants equally felt they could solve new problems regardless of the support they received during the worked example and practice problems and despite actual differences in problem solving performance.

3.5 Comparing results to previous study

Participants in the present experiment who received unscaffolded practice problems (and either hints or feedback during the study phase) had the exact same instructions as participants in

Margulieux and Catrambone (2019) who were in the groups (using Margulieux and Catrambone's terms), "guided constructive with hints and without feedback" and "guided constructive without hints and with feedback". Participants in these conditions received the same instructional materials, including worked examples and unscaffolded practice problems. The means from these conditions in both experiments were compared using a one-sample *t*-test, which compares data from the present experiment to a mean score from the literature, to ensure that the participants in each experiment performed equivalently and represent the same population. Participants who received hints but not feedback performed similarly in both studies, and the means were within the margin of error (Margulieux & Catrambone $M = 21.0$, present experiment $M = 19.5$, Std. error = 1.54). Likewise, participants who did not receive hints but received feedback performed within the margin of error (Margulieux & Catrambone $M = 21.5$, present experiment $M = 20.2$, Std. error = 1.54). Therefore, we conclude that the means of the groups who received unscaffolded practice problems are equivalent and that scaffolding practice problems with learner-constructed labels improves performance over this mean.

The quality of subgoal labels was also consistent across the two experiments. In Margulieux and Catrambone (2019), participants who received hints mainly constructed context-independent labels (69%) while relatively few of these participants constructed context-specific labels (22%) or incorrect labels (8%). In the present experiment, participants who received hints mostly constructed context-independent labels (75%) and again, relatively few participants constructed context-specific labels (22%) or incorrect labels (3%). Margulieux and Catrambone's participants who did *not* receive hints constructed fewer context-independent labels than those who received hints (45%). A larger portion constructed context-specific labels (36%) and incorrect labels (19%). Similarly, in the present experiment, participants who did not receive hints constructed fewer context-independent labels (35%), and more constructed context-specific (58%) or incorrect labels (7%). These similarities suggest that the learners, regardless of the experiment in which they participated, were comparable and that the instructions had the same effect on them. Therefore, the effect seems to be robust and repeatable.

4. Conclusions

Problem solving performance can be significantly improved by scaffolding initial problem solving with learner-constructed subgoal labels. Importantly for constructivist learning, learner-constructed labels, regardless of their quality, better supported initial problem solving than expert-constructed labels. Importantly for application, the scaffolding that improved problem solving was pre-determined for all learners, meaning that it can be incorporated directly into instructional materials and distributed at scale. The present study found no differences in the intervention's efficacy based on demographic differences, but this finding was based on a homogenous group of participants in terms of age, academic achievement, educational experience, and comfort with computers, which is not representative of the diverse array of learners who engage with many sources of programming, or problem solving more broadly, instruction. Though the present study replicated previous findings from Margulieux and Catrambone (2019), suggesting that the effect of the self-explanation intervention is stable, more research would be needed to test the scaffolding intervention in different groups.

The higher level of performance by learners who received their own subgoal labels as scaffolds provides further evidence for constructivist instructional strategies, even in independent learning environments without personalized feedback from peers or instructors. In this learning environment, learners constructed labels that were conceptually relevant and accurate enough that they could be applied as effective scaffolds to practice problems. Effective scaffolds help learners to apply procedural knowledge to novel problems (Pea, 2004), and if learner-constructed labels served as effective scaffolds, then they must be high quality. Implementing the scaffolds in problem solving instruction would be simple because the design is static, meaning that incorporating it would be no more time intensive than designing examples and problems to use in instruction.

The results of the present study also suggest that when learners construct subgoal labels with enough support, such as the hints or feedback provided in this study, they do not seem to benefit from expert-constructed subgoal label scaffolding. Scaffolding with experimenter-constructed labels did not improve problem solving in this experiment over unscaffolded practice problems. This finding, though surprising to the researchers, is not unprecedented and similar to findings from memory research on subjective organization. Memory research suggests that people recall more words when they organize the words themselves (i.e., subjective organization) than when they are told to use a prescribed organization (e.g., recall words in alphabetical order; Tulving, 1962). In this research, the words to memorize were unrelated to each other, and no method of organization is more or less correct than another. In the present research, however, there are correct and incorrect explanations and conceptualizations of the problem-solving procedure. Therefore, the finding learners' explanations improved performance more than an expert's explanations suggests that even though learners' explanation risk being incorrect or incomplete, they were more effective for scaffolding than an expert's explanation, despite the expert's explanation being developed through rigorous instructional design techniques.

Based on this study, we recommend that problem solving instruction promote self-explanation of the problem-solving process and then link those self-explanations to initial problem solving, such as through scaffolded practice problems. Our findings suggest that when learners are scaffolded with their own words rather than the words of experts, their later problem-solving performance improves. This instructional technique would engage learners in self-explanation, which promotes constructive knowledge building (Wylie & Chi, 2014), and helps connect learners' prior knowledge to initial problem-solving attempts. This connection could be especially important in learning environments in which there is limited instructional resources per student and in which learners have varied prior knowledge. Though the intervention was fixed, its use of self-explanation makes it personal to each learner, a characteristic that could be of great value for increasing learner independence.

References

Aleven, V. A. W. M. M., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science, 26*, 147-179.

Atkins, S. M., Sprenger, A. M., Colflesh, G. J. H., Briner, T. L., Buchanan, J. B., Chavis, S. E., … Dougherty, M. R. (2014). Measuring working memory is all fun and games: A four-dimensional spatial game predicts cognitive task performance. *Experimental Psychologist, 61*(6), 417-438.

Atkinson, R. K., Catrambone, R., & 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), 762-773.

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of the Educational Research, 70*(2), 181-214. doi:10.2307/1170661

Azevedo, R. (2005). Using hypermedia as a metacognitive tool for enhancing student learning? The role of self-regulated learning. *Educational Psychologist*, *40*(4), 199-209.

Azevedo, R., Moos, D. C., Greene, J. A., Winters, F. I., & Cromley, J. G. (2008). Why is externally-facilitated regulated learning more effective than self-regulated learning with hypermedia?. *Educational Technology Research and Development*, *56*(1), 45-72.

Catrambone, R. (1995). Aiding subgoal learning: Effects on transfer. *Journal of Educational Psychology, 87*(1), 5-17. doi:10.1037/0022-0663.87.1.5

Catrambone, R. (1996). Generalizing solution procedures learned from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*, 1020-1031. doi:10.1037/0278-7393.22.4.1020

Catrambone, R. (1998). The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General, 127,* 355-376. doi:10.1037/0096-3445.127.4.355

Catrambone, R. (2011, June). Task analysis by problem solving (TAPS): uncovering expert knowledge to develop highquality instructional materials and training. In *2011 Learning and Technology Symposium* (Vol. 47, pp. 132-139).

Chi, M. T. H. (2009). Active-constructive-interactive: A conceptual framework for differentiating learning activities. *Topics in Cognitive Science*, *1*(1), 73-105.

Conati, C., & VanLehn, K. (2000). Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education, 11,* 389-415.

Czerniewicz, L., Deacon, A., & Walji, S. (2018). Educators, copyright and open education resources in Massive Open Online Courses. In *Proceedings of the 11th International Conference on Networked Learning*. Bajić, M, Dohn, NB, de Laat, M, Jandrić, P & Ryberg, T. (Eds.). 264-271.

Downes, S. (2013). The role of open educational resources in personal learning. International Seminar of the UNESCO chair in e-Learning. Universitat Oberta de Catalunya.

Delen, E., Liew, J., & Willson, V. (2014). Effects of interactivity and instructional scaffolding on learning: Self-regulation in online video-based environments. *Computers & Education*, *78*, 312-320.

García Espinosa, B.J., Tenorio Sepúlveda, G.C. & Ramírez Montoya, M.S. (2015). Self-motivation challenges for student involvement in the Open Educational Movement with MOOC. *International Journal of Educational Technology in Higher Education, 12*, 91-103. doi:10.7238/rusc.v12i1.2185

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43.

Hill, J. R., & Hannafin, M. J. (2001). Teaching and learning in digital environments: The resurgence of resource-based learning. *Educational Technology Research and Development*, *49*(3), 37-52.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: a response to Kirschner, Sweller, and. *Educational Psychologist, 42*(2), 99-107.

Hundhausen, C. D., Farley, S. F., & Brown, J. L. (2009). Can direct manipulation lower the barriers to computer programming and promote transfer of training?: An experimental study. *ACM Transactions in CHI, 16*(3). doi:10.1145/1592440.1592442

Kim, M. C., & Hannafin, M. J. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice. *Computers & Education*, *56*(2), 403-417.

Kizilcec, R. F., Pérez-Sanagustín, M., & Maldonado, J. J. (2017). Self-regulated learning strategies predict learner behavior and goal attainment in Massive Open Online Courses. *Computers & Education*, *104*, 18-33.

Littlejohn, A., Falconer, I., & Mcgill, L. (2008). Characterising effective eLearning resources. *Computers & Education*, *50*(3), 757-771.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). *Programming by choice: urban youth learning programming with scratch* (Vol. 40, No. 1, pp. 367-371). ACM.

Margulieux, L. E., & Catrambone, R. (2019). Finding the best types of guidance for constructing self-explanations of subgoals in programming. *Journal of the Learning Sciences.* Published online 06/26/18. doi: 10.1080/10508406.2018.1491852

Margulieux, L. E., & Catrambone, R. (2016). Improving problem solving with subgoal labels in procedural instructions and worked examples. *Learning and Instruction, 42*, 58-71. doi: 10.1016/j.learninstruc.2015.12.002

Margulieux, L. E., Catrambone, R., & Guzdial, M. (2016). Employing subgoals in computer programming education. *Computer Science Education, 26*(1). doi: 10.1080/08993408.2016.1144429

Morrison, B. B., Decker, A., & Margulieux, L. E. (2016). Learning loops: A replication study illuminates impact of HS courses. In *Proceedings of the Twelfth Annual International Conference on International Computing Education Research* (pp. 221-230). New York, NY: Association for Computing Machinery. doi: 10.1145/2960310.2960330

Morrison, B. B., Dorn, B., & Guzdial, M. (2014). Measuring cognitive load in introductory CS: adaptation of an instrument. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, 131–138.

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, context, and worked examples in learning computing problem solving. *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 21-29). New York, NY: Association for Computing Machinery. doi: 10.1145/2787622.2787733

Ossiannilsson, E., Williams, K., Camilleri, A. F., & Brown, M. (2015). *Quality Models in Online and Open Education around the Globe: State of the Art and Recommendations*. International Council for Open and Distance Education (ICDE).

Palmiter, S., Elkerton, J., & Baggett, P. (1991). Animated demonstrations versus written instructions for learning procedural tasks: A preliminary investigation. *International Journal of Man-Machine Studies, 34,* 687-701. doi:10.1016/0020-7373(91)90019-4

Pea, R. D. (2004). The social and technological dimensions of scaffolding and related theoretical concepts for learning, education, and human activity. *Journal of the Learning Sciences, 13*(3), 423-451. doi: 10.1207/s15327809jls1303_6

Pirolli, P., & Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction*, *12*(3), 235-275.

Rohs, M. & Ganz, M. (2015). MOOCs and the claim of education for all: A disillusion by empirical data. *The International Review of Research in Open and Distributed Learning, 16*(6), 1-19.

Rountree, N., Rountree, J., Robins, A., & Hannah, R. (2004). Interacting factors that predict success and failure in a CSI course. *SIGCSE Bulletin, 33*(4), pp 101-104.

Saye, J. W., & Brush, T. (2002). Scaffolding critical reasoning about history and social issues in multimedia-supported learning environments. *Educational Technology Research and Development*, *50*(3), 77-96.

Schmidt, H. G., Loyens, S. M., Van Gog, T., & Paas, F. (2007). Problem-based learning is compatible with human cognitive architecture: Commentary on Kirschner, Sweller, and Clark. *Educational Psychologist, 42*(2), 91-97.

Smyth, R., Bossu, C., & Stagg, A. (2016). Toward an open empowered learning model of pedagogy in higher education. In Open learning and formal credentialing in higher education: curriculum models and institutional policies. AEMAL. IGI Publishing (IGI Global), Hershey, PA. United States, pp. 205-222.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, *29*(9), 850-858.

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, *22*(2), 123-138.

Tobias, S., & Duffy, T. M. (Eds.). (2009). *Constructivist instruction: Success or failure?*. Routledge.

Tulving, E. (1962). Subjective organization in free recall of “unrelated” words. *Psychological Review*, *69*(4), 344-354.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197-221.

Vygotsky, L. (1978). *Mind in Society.* Cambridge, MA: Harvard University Press.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, *17*(2), 89-100.

Wylie, R., & Chi, M. T. H. (2014). The self-explanation principle in multimedia learning. In R. Mayer (Ed.) *The Cambridge Handbook of Multimedia Learning, 2nd Edition (pp.413-432).* Cambridge University Press.

Yelland, N., & Masters, J. (2007). Rethinking scaffolding in the information age. *Computers & Education*, *48*(3), 362-382.

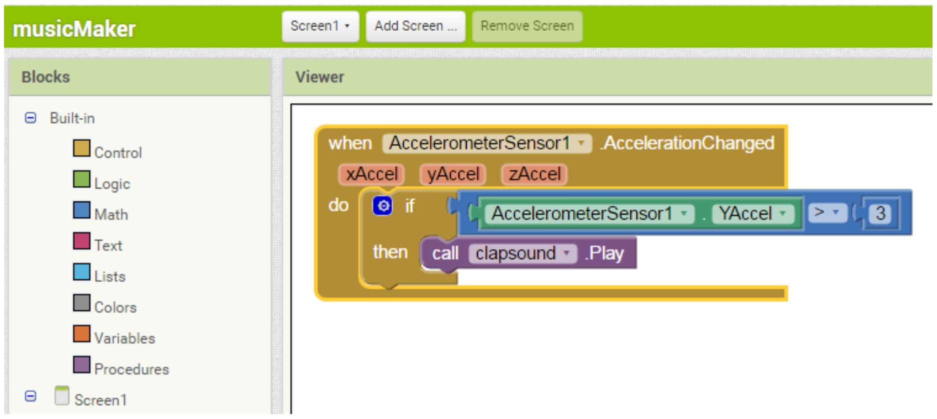
Unscaffolded Practice Problem	**Learner-Labeled Practice Problem**	**Expert Labeled Practice Problem**
Problem: Create an app that plays a cymbal sound when the image of a cymbal is touched.	Problem: Create an app that plays a cymbal sound when the image of a cymbal is touched. Subgoal 1: Subgoal 2:	Problem: Create an app that plays a cymbal sound when the image of a cymbal is touched. Handle Event: Set Output:

Figure 1. Unscaffolded practice problem compared to practice problems scaffolded by learner-constructed and expert-constructed subgoal labels.

Create Component
1. Click on the “Drawing and Animation” palette on the left.
2. Drag out a canvas to Screen1.

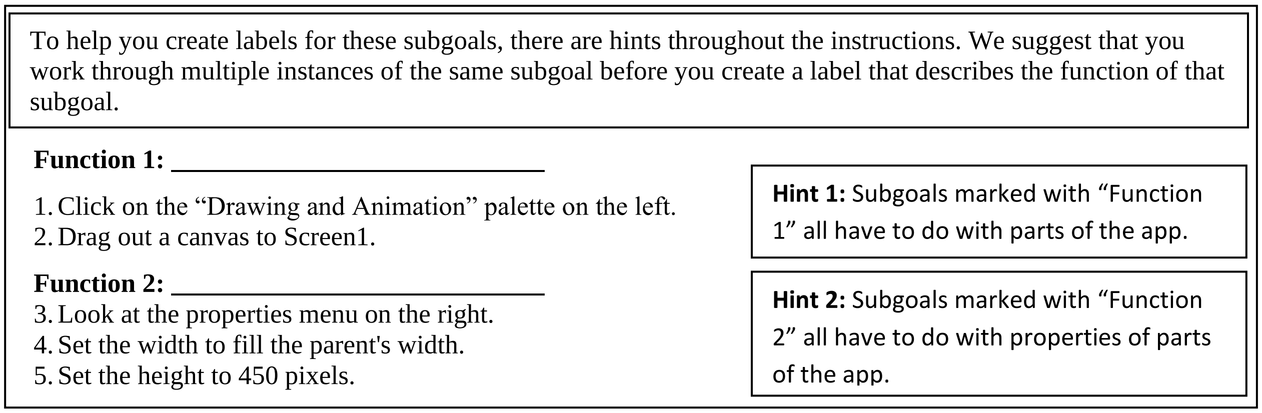
Set Properties
3. Look at the properties menu on the right.
4. Set the width to fill the parent's width.
5. Set the height to 450 pixels.

Figure 2. First several steps of the worked example of the procedure used to create the Music Maker app. In these steps, the user is creating the canvas on which to place the instrument images. Steps of the worked example are visually grouped into subgoals and labeled with meaningful subgoal labels that describe the function of that group of steps.



```
when AccelerometerSensor1 .AccelerationChanged
  xAccel  yAccel  zAccel
do  if  AccelerometerSensor1 . YAccel  >  3
    then  call clapsound .Play
```

Figure 3. App Inventor interface with interlocking blocks of code selected from menus used to program features. In this case, the clap sound will play when the phone detects a change in the y acceleration of the phone.



To help you create labels for these subgoals, there are hints throughout the instructions. We suggest that you work through multiple instances of the same subgoal before you create a label that describes the function of that subgoal.

Function 1: ______________________________

1. Click on the "Drawing and Animation" palette on the left.
2. Drag out a canvas to Screen1.

Function 2: ______________________________

3. Look at the properties menu on the right.
4. Set the width to fill the parent's width.
5. Set the height to 450 pixels.

Hint 1: Subgoals marked with "Function 1" all have to do with parts of the app.

Hint 2: Subgoals marked with "Function 2" all have to do with properties of parts of the app.

Figure 4. Worked example formatted for constructing subgoal labels with hints. The condition that did not receive hints were also told to work through multiple instances of the same subgoal before creating a label for it, but they did not receive the hints. Instead they were given feedback that included the expert-constructed subgoal labels, like in Figure 2.

Markers	Subgoal Identified by Expert	Hint
Function 1	Create components to be used in the app	Subgoals marked with “Function 1” all have to do with parts of the app.
Function 2	Set properties of components	Subgoals marked with “Function 2” all have to do with properties of parts of the app.
Function 3	Handle event/input to the app	Subgoals marked with “Function 3” all have to do with inputs from the user.
Function 4	Set output of the app	Subgoals marked with “Function 4” all have to do with outcomes of inputs from the user.
Function 5	Set conditions that moderate behavior	Subgoals marked with “Function 5” all have to do with conditions of inputs from the user.

Figure 5. Text included in the worked example for participants who received hints with their corresponding subgoal, as identified through task analysis. All hints also included the phrase “Please write the purpose of the subgoals marked with “Function X” in the blanks,” in which X was the number of marker in the worked example.

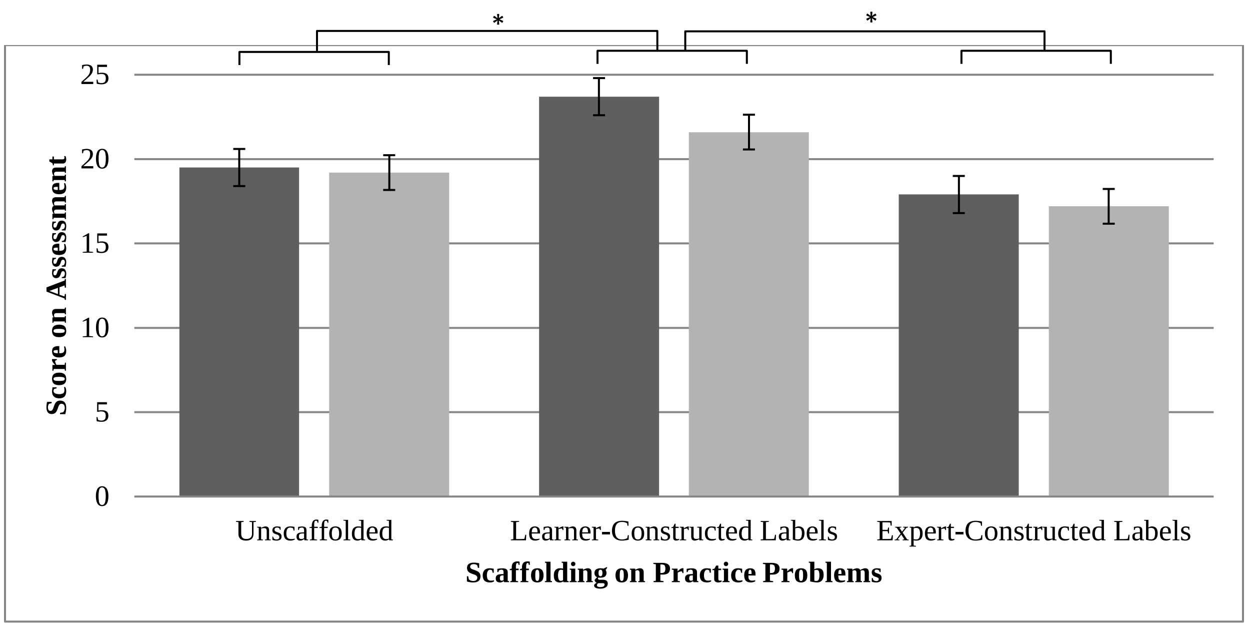


Figure 6. Performance on problem solving tasks among conditions. Dark bars are participants who received hints, and light bars are participants who received feedback. Maximum possible score was 25. Error bars are standard error. Statistically significant differences are indicated with asterisks.

Table 1

Demographic Averages for Participants and Their Correlation with Problem Solving Performance.

	Averages		Correlations	
	M	*SD*	*r or ρ**	*p*
Gender	58% male	-	.11*	.32
Age	19.6	2.2	.03	.74
Academic Major	62% engineering	-	.10*	.36
High School GPA	3.65	.26	.05	.64
Year in College	2.08	1.3	-.09	.41
College GPA	3.32	.45	.12	.27
Comfort with Computers (out of 7)	4.15	1.6	-.06	.56
Expected Difficulty (out of 7)	4.13	1.3	.04	.69
Score on Working Memory Task	1747 (normative mean = 1581)	411 (normative SD = 472)	.07	.51
Previous CS Courses	44% taken 1 course	-	.06*	.55

Table 2

Examples of Subgoal Labels Constructed by Participants for Each of the Coding Classifications.

Expert-Constructed Label	Context-Independent	Context-Specific	Incorrect
Create component	Add component to app	Create image sprite	Define variable
Set properties	Edit component	Name and add picture to image sprite	Select/drag
Handle input	Add interface command	Add condition for when clap is touched	Program functions
Set output	Set command result	Make clapsound play when clap is touched	Specify function
Set conditions	Add command conditions	Make something happen if the user moves the phone	New function

Table 3

Results for Intrinsic, Extraneous, and Germane Cognitive Load Measures.

	M	*SD*	*F* practice problem format	*p*	*F* subgoal learning method	*p*	*F* interaction	*p*
Intrinsic Load (out of 30)	9.8	6.3	1.87	.16	.34	.56	.05	.95
Extraneous Load (out of 30)	5.6	5.4	1.34	.27	.16	.69	.67	.51
Germane Load (out of 40)	24.4	7.5	.33	.72	1.41	.24	.28	.99