6-25-2018

Finding the Best Types of Guidance for Constructing Self-Explanations of Subgoals in Programming

Lauren Margulieux
Georgia State University, lmargulieux@gsu.edu

Richard Catrambone
Georgia Institute of Technology, richard.catrambone@psych.gatech.edu

Finding the Best Types of Guidance for Constructing Self-Explanations of Subgoals in Programming

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

[1]Georgia State University

[2]Georgia Institute of Technology

Author Note

The authors would like to thank Frank Durso, Mark Guzdial, Wendy Rogers, and Wendy Newstetter for critiques on earlier versions of the manuscript.

Correspondence concerning this article should be addressed to Lauren Margulieux, Learning Technologies Division, Georgia State University, Atlanta, GA 30302. Email: lmargulieux@gsu.edu

Abstract

Subgoal learning, a technique used to break down problem solving into manageable pieces, has been used to promote retention and transfer in procedural domains, such as programming. The primary method of learning subgoals has been passive, and passive learning methods are typically less effective than constructive methods. To promote constructive methods of learning subgoals, learners were prompted to self-explain the subgoals of a problem-solving procedure. Self-explanation asks learners to make sense of new information based on prior knowledge and logical reasoning. Self-explanation by novices is typically more effective when they receive guidance, because it helps them to focus on relevant information. In the present experimental study, the types of guidance that students received while self-explaining determined whether the constructive learning method was more effective than the passive method. Participants assigned to the constructive learning method performed best when they either received hints about the subgoals or received correct explanations as feedback, but not when they received both. These findings suggest that constructive learning of subgoals can further improve the benefits of subgoal learning when students receive only guidance that complements their construction of knowledge. This nuance is important for educators who engage their students in constructive learning and self-explanation.

Using Subgoal Learning and Self-Explanation to Improve Programming Education

Education has had a proliferation of resources that are intended to be used outside of a classroom to help students learn and practice problem-solving procedures. These resources can be used to augment a course, such as for a blended classroom, or used for self-directed learning, such as for a Massive Open Online Course (MOOC). While learners are using these resources, they are typically learning independently, meaning that they do not have immediate access to an instructor or peers. In procedural problem-solving domains, such as math and computer science, instructors and peers often provide key guidance that help students to resolve problem solving impasses (Newman, 1998; Roschelle & Teasley, 1995). To replace these common sources of guidance and make educational resources successful in procedural problem-solving domains, educators need to build-in additional support to guide students' learning. Useful tools for guiding learning include sophisticated software, such as intelligent tutoring systems, that adapt to the learner based on probability models of the learner's likely knowledge state (Polson & Richardson, 2013). These technologies, however, have long development times (Graesser, Hu, Nye, & Sottilare, 2016) and cannot reasonably accompany most educational resources, at least not until the resource has been adequately vetted or adopted to warrant the development cost. In the meantime, educational resources need a scalable solution to supporting students. The present research examined a low-tech, low-cost new strategy to support independent problem solving: the integration of subgoal learning and self-explanation.

This paper starts by reviewing the subgoal learning and self-explanation literature and explaining the design of instructional materials based on that literature. Then it describes the experimental research methods used to compare different versions of instructional materials and

discusses the quantitative and qualitative results of the research. Last, the paper summarizes the main findings and makes suggestions for future work.

Subgoal Learning

Subgoal learning refers to a strategy used predominantly in science, technology, engineering, and mathematics (STEM) fields that helps students to break down problem-solving procedures into manageable pieces, or subgoals. Subgoals are functional pieces of procedures used to solve problems. They are inherent in all procedures except the most basic. For instance, if algebra students were asked to solve the equation in Figure 1 for *x*, they would likely start by isolating terms with *x*s on one side of the equation and the others on the opposite side. Then they would simplify the terms until *x* had a coefficient of one. Isolating and simplifying terms are subgoals of the procedure used to solve for a variable. Novices have trouble recognizing the underlying function that steps serve and instead tend to focus on the individual steps taken to solve the problem (Bransford, Brown, & Cocking, 2000; Catrambone, 1998). Therefore, it is necessary to help students recognize the subgoals of problem-solving procedures by identifying subgoals directly in instructional materials (Catrambone, 1998).

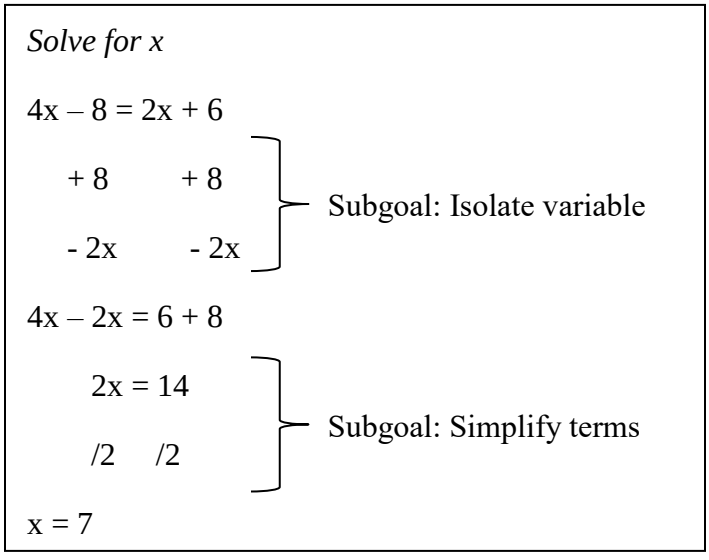

Figure 1. Worked example of the procedure used to solve for a variable. Steps of the worked example are grouped into the subgoals, denoted by brackets, necessary for solving problems in this class.

Past research has found that subgoal-oriented instructions help students to learn the subgoals of a procedure, causing them to better recognize the structural components of the problem-solving process (Atkinson, Catrambone, & Merrill, 2003; Catrambone, 1998; Margulieux, Catrambone, & Guzdial, 2016). By recognizing the structural components of the process, learners are more likely to correctly transfer their knowledge and apply the process to problems that used the same procedure but have different surface features or have modified or new individual steps (e.g., Catrambone & Holyoak, 1989). Better transfer of knowledge to solving new problems has been a consistent benefit of subgoal-oriented instructions across a variety of STEM domains, such as programming (e.g., Margulieux & Catrambone, 2016; Margulieux et al., 2016) and statistics (e.g., Catrambone, 1994, 1998).

Subgoal learning with worked examples. Worked examples are the most common type of subgoal-oriented instruction. Worked examples give learners concrete examples of a procedure being used to solve a problem. Because problems necessarily include concrete details, like solving the equation $x = 3 + 2$ rather than abstractly solving for a variable, worked examples include problem-specific information. Concrete details help learners to grasp the procedure before they can conceptually understand it (Atkinson et al., 2003). Eiriksdottir and Catrambone (2011) argued, however, that learning from worked examples does not promote deep processing of concepts. Although it may result in better initial performance because examples are more easily mapped to similar problems, it is less likely to aid retention and transfer than learning from abstract procedures (Eiriksdottir & Catrambone, 2011). Retention and transfer suffer when

learners study examples because novices tend to focus on surface features rather than structural features; surface features are easier to grasp, and novices do not have the domain knowledge to recognize the structural features of examples (Chi, Bassok, Lewis, Reimann, & Glaser, 1989).

To promote deeper processing of worked examples and, thus, improve retention and transfer, worked examples have been manipulated to promote subgoal learning. In particular, *subgoal labeling* is a technique used to promote subgoal learning and to help learners recognize the structure of procedures exemplified in worked examples (e.g., Catrambone, 1994, 1995, 1996, 1998). Subgoal labels are function-based instructional explanations that describe the purpose of a subgoal. For instance, for the problem in Figure 1 and for the subgoal in which the problem solver isolates terms with *x*s one on side the equation, the subgoal label might read "Isolate variable." This label provides information about the collective function of the individual steps within the subgoal.

Studies have found that receiving subgoal labels in worked examples improves performance on novel problems without increasing the amount of time learners spend studying instructions or solving problems (e.g., Margulieux et al., 2016). Subgoal labels are believed to be effective because they highlight the structure of examples, helping students focus on structural features and more effectively organize information (Atkinson, Derry, Renkl, & Wortham, 2000; Atkinson et al., 2003; Catrambone, 1995, 1996, 1998). By helping learners to focus on structural features of worked examples, subgoal labels are believed to reduce the extraneous cognitive load that is inherent in worked examples due to problem-specific details and can hinder learning (Renkl & Atkinson, 2002). Reducing extraneous cognitive load allows more mental resources to be devoted to learning the procedure through building schemata, chunking information, and connecting prior knowledge and new knowledge (Sweller, 2010).

Subgoal labeled worked examples are similar to process-oriented worked examples (e.g., Van Gog, Paas, & van Merrienboer, 2004). Both describe the purpose of steps in a worked example, but they are different in the abstraction of the explanation. Process-oriented examples explain each step of the solution and include the problem-specific details of the example in the explanation, meaning that an explanation can be used only for that step. On the other hand, subgoal labels explain the purpose of multiple steps and are independent from a specific problem-solving context, meaning that they can be applied to all problems of the same sort (Catrambone, 1995, 1998). Viewing multiple instances of each subgoal is critical to subgoal learning because it allows the learner to compare subgoals that achieve the same function but comprise different steps (Margulieux et al., 2016). For example, for the procedure in Figure 1, students should view multiple instances of isolating a variable so that they can compare different instances achieving the same function.

A main limitation of the implementations of the subgoal learning framework so far is that they have promoted passive learning by providing meaningful subgoal labels to learners rather than encouraging students to recognize the function of subgoals for themselves. This passive approach contradicts a growing body of evidence that learning is more effective when students actively or constructively engage with content rather than when they passively receive it. This body of evidence is summarized by Chi (2009) and used to support her Interactive-Constructive-Active-Passive (ICAP) framework. In this framework, Chi (2009) characterized four types of learning based on students' engagement with content: interactive, constructive, active, and passive (see Figure 2 for definitions and examples). Using this framework to compare the learning outcomes from various learning activities, Chi (2009) found that interactive was most effective, constructive was second-most effective, active learning was the third-most effective,

and passive learning was the least effective. The present study explored whether non-passive methods of learning subgoals would improve novel problem solving beyond the existing methods of learning subgoals. Because subgoal labels are, in essence, an instructional explanation of a problem-solving procedure, the present study explored the types of guidance that students would need to create explanations for themselves about the subgoals of a procedure.

	Passive	**Active**	**Constructive**	**Interactive**
Definition	Receiving information without activity	Receiving information with (usually physical) activity	Individually producing information beyond that which is provided	Collaboratively producing information beyond that which is provided
Examples	Listen to a lecture Read a textbook	Taking notes on a lecture Highlighting sections of a reading	Connecting concepts to prior knowledge Explaining the steps of a worked example	Discussing a concept Providing and responding to peer feedback

Figure 2. Definitions and characteristics of passive, active, constructive, and interactive learning based on the ICAP framework proposed by Chi (2009). Interactive learning was not represented in the present work.

Self-Explanation

Self-explanation is a common and effective learning strategy that could help students to learn subgoals. In self-explanation, students use prior knowledge and logical reasoning to make sense of new information and gain new knowledge. Most often, learners are constructing knowledge through this process and, thus, are engaging in a type of constructive learning (Bielaczyc et al., 1995; Chi et al., 1989; Schworm & Renkl, 2006). Sometimes, activities that encourage self-explanation provide so much guidance that self-explanation becomes a type of active learning (i.e., requires activity from the learner but not construction, per Chi's, 2009, definition). Active self-explanation typically involves learners selecting, rather than generating,

explanations from a list of possible explanations (e.g., Aleven & Koedinger, 2000; Conati & VanLehn, 2000). A review of self-explanation studies found that it is effective across a range of domains, if the domain has logical rules with few exceptions (Wylie & Chi, 2014). Self-explanation is commonly used with worked examples in procedural domains to improve learning outcomes (Wylie & Chi, 2014).

Similar to subgoal learning, self-explanation of worked examples identifies the features that are structural and reasons about the function of steps (Bielaczyc, Pirolli, & Brown, 1995; Chi, de Leeuw, Chiu, & LaVancher, 1994). Moreover, self-explanation is believed to be effective for many of the same reasons as subgoal learning. By self-explaining worked examples, learners recognize which features are structural and which are superficial. By recognizing the features that are most important, learners can reduce extraneous cognitive load devoted to processing surface features, allowing for more cognitive processes directed towards learning (Renkl & Atkinson, 2002). Self-explanation further improves learning processes because students tend to activate relevant prior knowledge as they think of possible explanations and integrate prior knowledge with new information to explore the plausibility of explanations (Chi et al., 1994; Sweller, 2010). These processes help learners build a better mental representation of the procedure that allows them to more easily apply their knowledge to novel problems (Renkl & Atkinson, 2003).

Constructive explanation is considered to have additional benefits over active explanation because it requires learners to generate an explanation. The generation effect states that learners remember information better when they produce it rather when they receive it (Jacoby, 1978). As deWinstanley, Bjork, and Bjork (1996) argued, the generation effect works because the cognitive processes involved in encoding information are similar to those involved

in retrieving information; therefore, the learner has the same cues while retrieving information as they had while encoding it. In their review of self-explanation literature, Wylie and Chi (2014) found that constructive self-explanation was more effective than active self-explanation, but few learners engage in constructive explanation without prompting.

Prompting self-explanation. Learners do not commonly engage in constructive self-explanation on their own (Chi et al., 1989; Renkl, 2005). Chi et al. (1989) found that about 10% of learners self-explained examples without external prompting. Many studies have replicated this low rate of self-explanation or found that even fewer learners generate their own explanations and instead paraphrase others' explanations (e.g., Hausmann & Chi, 2002). Renkl and colleagues (1998, 2005) argued that many learners do not self-explain, especially when they have little prior knowledge, because it requires a large amount of effort and mental resources. When prompted to self-explain, though, most learners can successfully generate explanations if they devote additional time to the task (Wylie & Chi, 2014).

Prompted self-explanation leads to the same learning outcomes as intrinsically motivated self-explanation, suggesting that self-explanation itself leads to learning benefits rather than characteristics of students who self-explain (e.g., Bielaczyc et al., 1995; Chi et al., 1994; Hausmann & Chi, 2002). To encourage consistent self-explanation, Renkl, Stark, Gruber, and Mandl (1998) found that instructions needed to include prompts throughout. Prompts range in the amount of guidance that they provide. Completely open-ended questions, like "Can you explain that?" (Hausmann & Chi, 2002), provide no guidance. Focused questions, like "Explain how examples 1 and 2 are similar," (de Koning, Tabbers, Rikers, & Paas, 2011), direct learners' attention and, thus, provide some guidance. Prompts that provide a lot of guidance and can result in active rather than constructive self-explanation range from filling in blanks of partial

explanations (Berthold, Eysink, & Renkl, 2009), to selecting explanations from a menu (Conati & VanLehn, 2000).

Which self-explanation prompt is most effective depends on the learner's prior knowledge; the amount of information in the prompt needs to be balanced with the learners' knowledge (Renkl, 2002). In their review of the self-explanation literature, Wylie and Chi (2014) found that self-explanation is not effective if learners do not have gaps in their understanding after instruction. In other words, if learners are given all the knowledge that they need, nothing is left to construct. Therefore, prompts should not include so much information such that self-explanation is not necessary. For example, if a prompt stated, "Problems 1 and 2 are similar because they both use Newton's second law. Explain how problems 1 and 2 are similar.", then the learner would not have the opportunity to recognize the underlying law that gives both problems a common structure. If learners are given too little information, however, they spend too much of their cognitive capacity trying to figure out what they should be learning to actually learn (Kirschner, Sweller, & Clark, 2006). For example, Wylie and Chi (2014) found that focused self-explanation prompts, such as "Could you explain how problems 1 and 2 are similar?" were typically more effective than completely open-ended prompts, such as "Could you explain the problems?" They argued that novices know so little about domains that they need clues about what to explain to be most effective. In the present study, the amount of information provided in prompts was varied to explore the effect of this type of guidance on self-explanations of subgoals.

Feedback on self-explanation. In a recent meta-analysis of types of feedback in computer-based learning environments, Van der Kleij, Feskens, and Eggen (2015) discussed the merits of three types of feedback: 1) knowledge of response, which tells the learner only whether

their answer is correct or not, 2) knowledge of correct response, which tells the learner the correct answer, and 3) elaborated feedback, which tells which answer is correct and elaborates on why it is correct, often providing additional instruction in the feedback (Van der Kleij et al., 2015). In their review, they found that knowledge of correct response leads to better learning than only knowledge of response, and in most cases, elaborated feedback leads to better learning than knowledge of correct response (Van der Kleij et al., 2015).

Elaborated feedback is by far the most common type of feedback in constructive learning environments (Molloy & Boud, 2014; Thurlings, Vermeulen, Bastiaens, & Stijnen, 2013). Though it requires more time and labor than other types of feedback (Jaehnig & Miller, 2007), it contributes to an effective feedback loop between learner and teacher in face-to-face constructive learning environments (Thurlings et al., 2013). The educational technology field is working to replicate this feedback loop between learners and technology, such as with intelligent tutoring systems (e.g., Kulik & Fletcher, 2016; Ma, Adescope, Nesbit, & Liu, 2014), but the time-intensive development process makes personalized, elaborated feedback inaccessible for many educational resources. Because the present study aimed to inform the development of scalable, online educational resources, part of its purpose was to examine the impact of knowledge of correct response feedback on constructive learning.

The literature suggests that whether learners should or should not receive correct response feedback depends on characteristics of the task and learners. In some situations, learners who compare their self-explanations to a correct explanation (i.e., correct response feedback) perform better than those who do not. For instance, Renkl (2002) found that learning outcomes improved when participants who self-explained the worked examples could also access short instructional explanations to check their self-explanations compared to participants

who could not access explanations. Renkl (2002) argued that the instructional explanations were necessary to reduce illusions of understanding and keep learners from perpetuating incorrect explanations.

In other situations, however, access to correct explanations can hinder performance. For instance, Schworm and Renkl (2006) found that when learners were prompted to self-explain, learning outcomes were better without access to correct explanations than with access to them. Schworm and Renkl (2006) argued that learners overly relied upon the correct explanations and would not devote much effort to constructing self-explanations before seeking the correct explanations provided by the feedback. For these reasons, Schworm and Renkl (2006) suggested that withholding correct explanations from learners might be more beneficial in some cases than ensuring that learners' self-explanations are correct.

In summary, self-explanation and subgoal learning both support effective organization of information and direct cognitive resources to structural features of worked examples of procedural problem solving. The present study explored whether self-explanation could provide an effective active or constructive method of learning subgoals. In addition, the present study explored whether subgoals created by an instructional designer could provide effective correct response feedback to learners who are actively or constructively learning subgoals.

Present Study Instruction

The main research question for the present study was, "What is the most effective method for students who are studying worked examples to learn subgoals for well-defined problem-solving procedures?" An additional constraint on this question was that the only feedback students had access to was correct response feedback. To address this research question within

this context, participants were prompted to learn the subgoals of a procedure through a worked example that either encouraged passive, active, or constructive learning. The study also manipulated whether participants received correct response feedback on the subgoals that they selected (active method) or created (constructive method). A secondary research question for participants who learned constructively was, "What kind of subgoal labels do students construct when given different types of guidance?"

Learning environment. The problem-solving domain for the present study was programming. Programming is a procedurally-focused STEM field that typically includes worked examples and practice problems in instruction. The acquisition of programming skill has been facilitated by self-explanation of goals and procedural structure (Soloway, 1986; Pirolli & Recker, 1994) and subgoal learning (Margulieux et al., 2016; Margulieux & Catrambone, 2016), so it was an appropriate domain to test the interventions.

To control for prior knowledge, participants were required to have little programming experience. Because participants were novices, the present study used a drag-and-drop programming language to teach programming concepts. Drag-and-drop programming languages are more easily understood by novice learners because they can select and drag pieces of code from a menu, which does not require learning the syntax and semantics of a programming language (Hundhausen, Farley, & Brown, 2009; see Figure 3). The programming language used in the present study was Android App Inventor, which is used to create applications (apps) for Android devices. Participants used App Inventor to create an app that has buttons that play sounds when pressed.

Figure 3. App Inventor interface with interlocking pieces of code selected from menus used to program features.

Participants came into a computer lab reserved exclusively for data collection to ensure that they did not have outside help or distractions. Each participant sat at a desk with a computer and completed the study independently, meaning that they did not receive help from fellow participants or the experimenter. Up to four participants could complete the study at once. Participants used the computer to access the computer-based working memory test, to watch a video that gave an overview of App Inventor, and to use the App Inventor interface. The other instructional materials were given to participants in paper form for a few reasons. First, paper instructions allowed participants to use the computer for interacting with App Inventor without having to rotate through various programs. Second, participants were encouraged to take notes on the paper instructions, especially related to creating subgoal labels. Lastly, participants gave the paper instructions back to the experimenter before starting the assessment tasks, ensuring that they did not have access to the instructional materials during the assessment.

Instructional material manipulations. All instructional manipulations were in the worked example that participants received. The worked example listed the steps taken to create a

Music Maker app that plays musical sounds when images of instruments are pressed or the device is tilted. For instance, a drum sound would play when a drum image is pressed, or a tambourine sound would play when the phone is tilted. An excerpt of the example, in all passive, active, and constructive formats, can be seen in Figure 4. The excerpt shows only two of the five subgoals, create component and set properties, that were demonstrated in the worked example (all five can been found in Table 1). The subgoals of the procedure were identified using the Task Analysis by Problem Solving (TAPS) procedure (Catrambone, 2011) that has been used in prior research (e.g., Margulieux & Catrambone, 2016). The format of the worked example depended on participants' assigned method of subgoal learning.

In the passive learning condition, participants were given subgoal labels created by the experimenters, as is conventional in prior subgoal research (e.g., Catrambone, 1998; see Figure 4a). These subgoal labels were also created through the TAPS procedure (Catrambone, 2011).

a. Passive Condition Excerpt

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.

b. Active Condition Excerpt

This procedure has five subgoals, which are listed below. **As you create the app, please match these subgoal labels with the "function" blanks provided.** All of the functions that are the same number, are the same subgoal. For example, all sections labeled with "function 1" achieve the same subgoal.

Create components, Set properties, Handle input, Set output, Set conditions

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.

c. Guided Constructive with Hints Excerpt

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.

d. Guided Constructive without Hints was the same but without hints

e. Unguided Constructive Excerpt

This procedure has five subgoals. **As you create the app, please group the steps of the procedure into subgoals. Then create subgoal labels that describes the purpose of the subgoals that you've created.**

1. Click on the "Drawing and Animation" palette on the left.
2. Drag out a canvas to Screen1.
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 4. Worked example formatted for passive (a), active (b), and constructive (c, d, and e) conditions.

Table 1

Experimenter-Created Labels (with how often each occurred in the worked example) and Examples of Subgoal Labels Constructed by Participants for Each of the Coding Classifications.

Subgoals as Identified by Experimenter	Participant-Created Labels				
	Problem-Specific	Higher-Level Problem-Specific	Problem-Independent	Hint-Term Problem-Independent	Incorrect
Create component (occurred 8 times in example)	Create image sprite	Create a canvas that fills the screen	Add component to app	Begin new object	Define variable

Set properties (occurred 7 times in example)	Name and add picture to image sprite		Edit component	Add properties to app	Select/drag
Handle input (occurred 4 times in example)	Add condition for when clap is touched	Make a sound when clap icon is touched	Add interface command	Set user inputs	Program functions
Set output (occurred 6 times in example)	Make clapsound play when clap is touched		Set command result	Set outcomes of inputs	Specify function
Set conditions (occurred 3 times in example)	Make something happen if the user moves the phone	When the phone is tilted down, the clap sound will play	Add command conditions	Establish input conditions	New function

In the active learning condition, participants were given the worked example grouped by subgoals and asked to select a subgoal label from a list of labels that matched the purpose of the group (see Figure 4b). The list contained only labels that were viable options, meaning the list did not include distractor items that were not applicable to the procedure being learned (see left column of Table 1 for list of subgoal labels). Requiring novice learners to distinguish between explanations that might or might not apply to the procedure would likely have unnecessarily added to the cognitive load required to complete the task. Participants could have completed this activity incorrectly, though, by selecting the wrong label for a group of steps. This active method of self-explaining was similar to the active self-explanation methods used by Aleven and Koedinger (2002) and Conati and VanLehn (2000).

In the constructive learning conditions, participants were asked to create their own subgoal labels to explain the subgoals of the procedure. To train participants to construct their

own labels, they were given subgoal label training. Only the constructive groups received this training. Groups who do not receive constructive learning conditions should not receive constructive training because it might prompt them to use constructive learning methods during the study, which could confound the results. Instead, the passive and active groups received a comparable task: analogy training. Training for analogies (e.g., water : thirst :: food : hunger) was considered similar because both analogies and subgoal labeling require people to consider the underlying relationship between words and come up with a new word that describes that relationship. In the present study, self-explanations were written onto the paper-based worked example that participants studied. Schworm and Renkl (2006) found that written self-explanations are better than spoken explanations because they require articulating thoughts and creating a record, which allows students to reflect on their explanations more easily.

Three constructive learning conditions prompted participants to construct their own subgoal labels with different types of guidance. There were two types of guided constructive conditions in which participants were given the worked example with the solution steps already grouped by subgoal, and the example indicated which subgoals achieved the same functions. For instance, all the subgoals denoted as “Function 1” achieve the same function though the exact steps taken were different (see Figure 4c and 4d). Note that the term function is used instead of subgoal on the worked example for participants because the non-technical meaning of function was more descriptive of the desired output than the non-technical meaning of subgoal. Subgoals appeared between three and eight times in the worked example. See left column of Table 1 for the full list of subgoals and the number of times that they appeared.

In the guided constructive with hints condition, participants were given hints about the similarities among different instances of the same subgoal (see Figure 4c). In the guided

constructive without hints condition, participants did not receive hints (see Figure 4d). In the unguided constructive condition, participants received a worked example that did not indicate which steps belonged to which subgoals (see Figure 4e). Participants in this condition had to identify the subgoals for themselves and create labels for them.

The type of guidance that participants received during instruction also differed based on whether they received correct response feedback. Instructions for participants who received this type of feedback had another copy of the worked example that included subgoal labels created by the experimenters. For the passive condition, this copy was the same as the initial worked example. For the active and constructive conditions, the copied example with experimenter-created subgoal labels provided correct response feedback to the participants. Participants who received feedback were asked to compare their labels to those created by the experimenter. Instructions for participants who did not receive feedback included only the worked example with the passive, active, or constructive interventions. These participants were asked to re-read the example to make time on task more similar to that of participants who received feedback, as is common in the self-explanation literature (e.g., Chi et al., 1994). The exception was that participants in the passive and no feedback condition were not asked to re-read the example to make their experience different from those in the passive with feedback condition. Due to this difference, the time on task was different, providing some insight into how time on task affects performance.

Because the worked example was long, participants received only one worked example. Giving one worked example provided a rare opportunity to ensure that participants in the feedback condition did not overly rely on feedback. Participants were not told that they would

receive feedback until they completed the task, meaning that they did not know to expect feedback.

Hypotheses. The guidance provided by correct response feedback was expected to interact with subgoal learning methods: passive, active, guided constructive with hints, guided constructive without hints, and unguided constructive. The subgoal learning methods are listed in order from providing the most information about the subgoals (i.e., passive gives participants all the information about the subgoals of the procedure) to the least information (i.e., unguided constructive provides no information about the subgoals of the problem, only training for making subgoal labels). The more information that is provided for the learners, the less information that learners can construct for themselves (Wylie & Chi, 2014). Conversely, the less information that self-explanation prompts provide, the more likely learners are to flounder because they must recognize connections between pieces of information that are not necessarily apparent to novices (Wylie & Chi, 2014). In the case that the prompt did not provide sufficient information, correct response feedback was expected to help learners who had struggled to create self-explanations. Therefore, learners who received less information from prompts were expected to create worse self-explanations and benefit from the extra information provided by feedback. Feedback, however, was expected to be unnecessary for learners who received more information while constructing self-explanations. The guided constructive with hints condition, which explicitly draws connections between analogous subgoals (see hints in Figure 4c), was expected to promote a mental organization of information that fostered insight, making correct response feedback in this condition unnecessary.

Method

Design

The experiment manipulated two variables. Subgoal learning method (passive, active, guided constructive with hints, guided constructive without hints, or unguided constructive) was crossed with correct response feedback (no feedback or feedback) to create a total of 10 groups. An experimental design conducted in a controlled laboratory setting was chosen for this stage of research so that the effects of the manipulations could be isolated from other factors of the learning environment and causal relationships could be drawn between variables and outcomes. The design was between-subjects, meaning that each participant was randomly assigned to only one of the 10 groups. Learning outcomes were measured by performance on problem-solving tasks, performance on explanation tasks, and time on task for the assessments and for the instructional period. Demographic characteristics, working memory capacity, pre-test and post-test score, subjective cognitive load, and perception of understanding were also collected as possible predictors of performance. The subgoal labels that participants construct were collected and analyzed for content. Quality of subgoal label was also considered as a possible predictor of performance.

Participants

Each of the 10 conditions had 25 participants (N = 250). Participants were students at a mid-sized, technical college in the Southeastern United States and recruited through course credit in psychology classes. Participants did not have prior experience with Android App Inventor and did not take more than one high school or college-level course in computer science or computer programming. These limitations were necessary because instructional materials were designed for novices.

Pre-instruction procedure and materials

Sessions took between 80 and 110 minutes, depending on how quickly participants completed each of the tasks. An overview of the procedure can be found in Figure 5. First, participants completed the demographic questionnaire, working memory measure, and pre-test. Demographic information was collected for participants' age, gender, academic field of study, high school GPA, college GPA, year in school, computer science experience, comfort with computers, and expected difficulty of learning App Inventor because they are possible predictors of performance (Rountree, Rountree, Robins, & Hannah, 2004). These demographic characteristics were not found to correlate with problem solving performance (see Table 2).

Procedure		
Pre-Instruction 10-15 minutes	Demographic questionnaire	
	Working memory measure	
	Pre-test	
Instructions 40-55 minutes Participants had access to all instructions throughout this entire period.	Overview video	
	Subgoal label training for all constructive groups	Analogy training for passive and active groups
	Worked example – same content for all groups but different format based on subgoal learning manipulation as shown in Figure 3.	
	Feedback for feedback groups	Re-read example for no feedback group
	Practice problems – same for all groups	
Post-Instruction 30-40 minutes During this period, participants had access to App Inventor interface but not written instructions.	Cognitive load measure	
	Post-test (learning check)	
	Assessment 1: Problem solving tasks (max. 25 minutes)	
	Assessment 2: Explanation tasks	

Figure 5. Overview of procedure. All materials are the same among conditions unless otherwise noted.

Table 2

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

	Averages		Correlation	
	M	*SD*	*r*	*p*
Gender	62% male	-	.06*	.39
Age	19.6	2.6	-.06	.34
High School GPA	3.88	.24	-.002	.98
Year in College	2.13	1.3	.03	.60
College GPA	3.40	.48	-.006	.95
Comfort with Computers (out of 7)	4.08	1.6	.12	.06
Expected Difficulty (out of 7)	4.11	1.3	.11	.09
Previous CS Courses	58% taken 1 course	-	-.001*	.98

Note: Correlations marked with * are point biserial correlations due to one variable being dichotomous.

Participants' working memory capacity was measured because previous research has found that working memory capacity predicted success at self-explanation (Wylie & Chi, 2014). The Shapebuilder task was used to measure working memory capacity (Atkins et al., 2014). The Shapebuilder task is a four-dimensional task that includes a four-by-four grid and four sets of shapes (i.e., square, circle, diamond, and triangle) in different colors. The computer-based task presents to the participant a sequence of colored shapes on the grid, and the participant is asked

to match the order, location, shape, and color of the items presented (Atkins et al., 2014). This task is similar to the problem-solving procedure of creating an app because both involve dragging items of various shapes and colors in a particular order to correctly achieve the task; therefore, it was considered an appropriate tool for measuring working memory capacity for this procedure.

Please answer the following questions about Android App Inventor.
If you don't know the answer, please mark "I don't know." There is no penalty for this.

1. To create an ImageSprite component for your app, drag out the ImageSprite from _____.
    a. I don't know
    b. User Interface
    c. Sensors
    d. Drawing and Animation
2. You add animations, such as ImageSprites, to your app, you need a canvas.
    a. I don't know
    b. True
    c. False
3. To play a sound, you need a block called…
    a. I don't know
    b. "set *sound* source to"
    c. "call *sound* play"
    d. "*sound*"
4. To interact with an ImageSprite, you need a block called "when *ImageSprite* _____."
    a. I don't know
    b. Touched
    c. Clicked
    d. Pressed
5. To use a change in a phone's accelerometer in the app, you use a block called "when *AccelerometerSensor* AccelerationDifferent."
    a. I don't know
    b. True
    c. False

Figure 6. Instruction and items on pre-test and post-test.

Participants completed a multiple-choice pre-test to ensure that they were truly novices of App Inventor. The five pre-test questions asked about the most basic App Inventor features to capture any rudimentary knowledge that participants had (see Figure 6). For each question in the

pre-test, one of the answer choices was “I don’t know” to avoid forcing participants to guess and introducing unnecessary error. The majority of participants (89%) scored a zero on the pre-test, and no participants scored higher than one point.

Instructional procedure and materials

After the pre-instruction period, participants started the instructional period (see Figure 5). All manipulations occurred within the instructional period. The instructional period started with an overview video of the App Inventor interface that was the same for all participants. The purpose of this video was to introduce participants to the App Inventor interface and the types of tasks that can be completed with App Inventor. The video did not include information about the procedure being taught, but it was intended to help participants familiarize themselves with the problem-solving space in which they would be working.

After the introductory video, participants received the paper instructions, starting with either subgoal label or analogy training. The subgoal label training was given only to participants in the three constructive conditions to avoid inadvertently prompting participants in the passive and active conditions to construct subgoal labels and confounding results. To control for time on task, participants in the passive and active conditions were given analogy training instead, which had the same structure as the subgoal label training. The subgoal label training defined subgoals and explained their benefits, gave an example of subgoals (similar to Figure 1), asked participants to make subgoals for an order of operations problem, and asked them to compare their labels to those made by an expert (for full training see supplemental online material).

Next, participants received the Music Maker worked example. They were randomly assigned to one of the five subgoal learning conditions (see Figure 4). When participants finished the first pass through the worked example, they were either prompted to re-read the example for

the no feedback condition, or they were given the worked example with the experimenter-created subgoal labels for the correct-response-feedback condition (see left column of Table 1). Participants in the feedback condition were told that the subgoal labels in the second copy of the worked example were created by a subgoal label expert. Then they were asked to compare the labels that they made or selected to those given in the second example. In the passive condition, the initial worked example that they received already included the experimenter-created subgoal labels. Therefore, the worked example was the same during the instruction and feedback stages. To make the passive no feedback and passive feedback conditions different, participants in the no feedback condition were not asked to re-read the example. This difference provided some insight into the effect of re-reading the example and time on task.

To ensure that participants paid attention to the worked example and could complete tasks in the App Inventor interface, they were asked to successfully complete practice problems before starting the assessment period (see Figure 7). Of the four tasks that participants completed, two required isomorphic transfer from the worked example, meaning that they used the same procedural steps as the worked example and differed only in surface features. For instance, the worked example showed the steps to create a drum image that plays a drum sound when pressed, and an isomorphic transfer practice problem asked participants to create a cymbal image that plays a cymbal sound when pressed. The other two practice problems required contextual transfer, meaning that they used the same procedural steps as the worked example and differed in surface and contextual features. For instance, if the example showed steps to create a drum that plays a sound when pressed, a contextual transfer practice problem asked participants to create an accelerometer sensor that plays a sound when the phone is tilted down.

Complete the following tasks in the App Inventor website. Try to complete them without looking at previous instructions, but if you need to refer to the instruction, you can.

1. Add a cymbal image and cymbal sound to the app and set the source files to cymbal.gif and cymbal.wav
2. Program the cymbal sound to play when the cymbal image is touched.
3. Program the cymbal sound to play when the phone bottom is tilted down (negative YAccel value)
4. Challenge question: Program the "clap" ImageSprite to move 5 pixels to the right of its current location when it is touched (hint: components called "ImageSprite.X" deal with the x coordinates of an ImageSprite).

Figure 7. Practice problem solving tasks given at the end of the instructional period. The page given to participants had spaces between the problems so that participants could write notes.

Assessment procedure and materials

When participants finished the practice problems, they were asked to return all the paper instructions that they had received. They still had access to the App Inventor interface and the work that they had done during the instructional period. To measure cognitive load experienced during the instructional period, participants completed a questionnaire for measuring cognitive load induced during programming instruction that was developed by Morrison, Dorn, and Guzdial (2014). This questionnaire was given directly after the instructional period to measure cognitive load experienced during instruction and not the cognitive load experienced during assessment. The questionnaire included three questions about intrinsic cognitive load (i.e., the load associated with processing information that is necessary to learn the procedure; e.g., "The topics covered in the activity were very complex,"), three questions about extraneous cognitive load (i.e., the load associated with processing information that is not necessary to learn the procedure; e.g., "The instructions and/or explanations during the activity were very unclear,"), and four questions about germane cognitive load (i.e., the load associated with cognitive

processes of learning; e.g., “The activity really enhanced my understanding of the programming concepts covered,”).

Following the cognitive load questionnaire, participants took a post-test that contained the same items as the pre-test about the most basic features of App Inventor. The post-test served as a learning check to ensure that participants had learned to use App Inventor. The majority of participants (81%) scored the full five points on this post-test, and no participants scored lower than four points. Therefore, no participants were excluded from the analyses based on post-test score. Participants were also asked to rate how well they understood the instructions and how comfortable they would be solving novel problems using the procedure.

After these checks, participants completed assessment tasks that measured problem solving knowledge. The first set of assessment tasks was problem-solving tasks that asked participants to modify or add components to their Music Maker app. Of the five problem-solving tasks, two required contextual transfer (i.e., nearer transfer) from the worked example. For contextual transfer problems, the surface features of the app components were different, but the procedural steps used to create them were the same (e.g., the steps used to create a cymbal component and create a drum component are the same except for the file used). One of these assessment questions was, “When playing long sound clips instead of short sounds clips, it’s better to use the player component than the sound component. Write the steps you would take to add a melody to your Music Maker app and make it play when touched. Create a new ImageSprite for the melody.” This task followed the same steps as the worked example to add instruments to the app, but using a player component instead of a sound component. The remaining three tasks required procedural transfer (i.e., farther transfer) from the worked example, meaning that the individual steps used to create the app components were different, but

the procedure used to create them was structurally the same. One of these assessment questions was, “Write the steps you would take to create a new app using cowbell.jpg and cowbell.wav so that the sound played when the picture was touched or when the phone is shaken.” The procedure used to create the new cowbell app was the same as the procedure to create the Music Maker app from the worked example, but the steps taken to do so were different. Because the cowbell app required a sound to play when the phone was shaken in addition to an ImageSprite being touched, participants had to use new series of steps to solve the problem, but they were achieving the same subgoals as they had for the Music Maker app.

Participants were asked to attempt the tasks in the App Inventor interface and then to write down the steps that they took so that their problem-solving process could be scored. In the interface, completing later steps of a task can rely on correctly completing earlier steps of a task. For instance, a participant could not program a sound to play when an image is clicked if they could not create the image. For this reason, participants were asked to write down steps that they would take to complete the task, even if they could not complete them in the interface. Asking participants to write their solution also shows the steps that they took to reach the solution rather than only the result. Participants wrote their solutions in an open-ended format, but multiple scorers were not necessarily because the procedural problem-solving steps were easy to identify as correct or incorrect. Participants had up to 25 minutes to complete the problem-solving tasks.

In addition to measuring problem solving performance, a second set of assessment tasks was used to measure whether participants could recognize the function that a step of a solution serves. This set of tasks was intended to measure participants’ knowledge of the procedure, regardless of whether they could correctly apply it to solving a novel problem. Thus, participants received solutions to problem-solving tasks and were asked to match each step of the solution to

the subgoal label that correctly explained the function of that step by drawing a line between the steps and subgoal labels. A matching task was used rather than a more open-ended task to compare participants' knowledge to the subgoal structure identified through task analysis. The task instructions stated that each step could only be matched to one subgoal label, but each subgoal label might be the correct explanation for multiple steps. Participants were given the solutions to the problem-solving tasks that they had just attempted to solve to make the problem solving and explanation tasks more congruous and reduce the amount of new, surface information that participants needed to process.

Results and Discussion

Guided constructive learning improved problem-solving performance

For the problem-solving assessment, participants received a score for number of correct steps taken towards problem solutions. Because the tasks involved numerous steps, scoring based on steps provided more sensitivity than scoring based on whole answers. The maximum possible score was 25. The total average mean was 18.26, and the total average standard deviation was 5.08. Performance on the problem-solving tasks depended on the interaction of subgoal learning method and correct response feedback, $F(4, 240) = 4.91$, $MSE = 21.9$, $p = .001$, partial $\eta^2 = .076$ (see Figure 8). Due to the disordinal nature of this interaction, the main effects will not be reported to avoid confusion in interpreting the results (Maxwell & Delaney, 2004). To explore this interaction and determine the effect of correct response feedback on each method of learning subgoals, simple main effects comparisons were used. This analysis found that feedback affected performance only for the guided constructive groups, but it affected them in different ways (see Table 3). Participants in the guided constructive with hints conditions performed statistically better when they did not receive correct response feedback than when they did, whereas

participants in the guided constructive without hints conditions performed statistically better when they received feedback than when they did not. Thus, people who received one type of support, either hints or correct response feedback, performed better than those who received both or neither types of support.

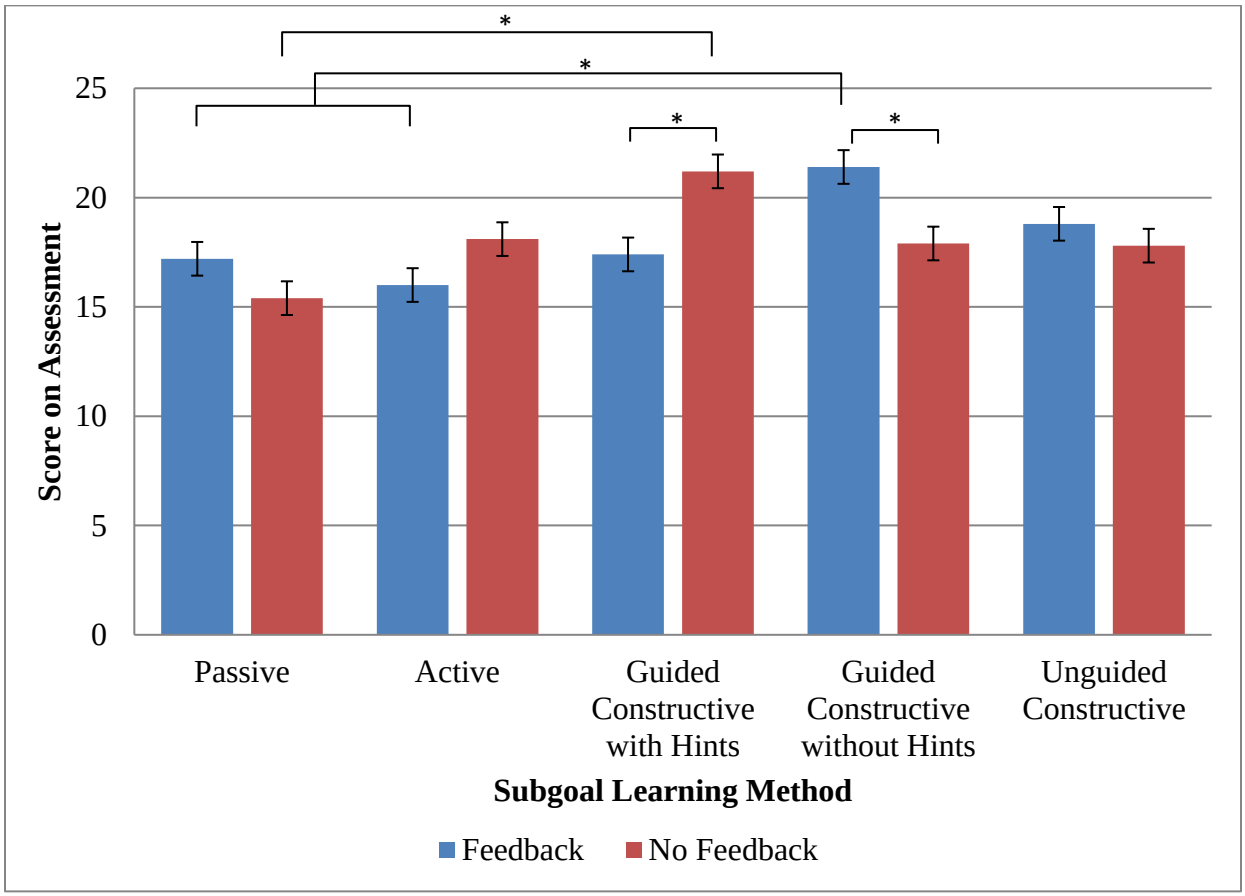

Figure 8. Performance on problem solving tasks among conditions. Maximum possible score was 25. Error bars are standard error. Statistically significant differences are indicated with asterisks.

Table 3

Simple Main Effects Analysis of Feedback on Problem Solving Performance.

Subgoal Learning Method	Mean for No Feedback	Mean for Feedback	Mean Difference	Std. Error	*p*

Passive	15.4	17.2	-1.80	1.32	.175
Active	18.1	16.0	2.08	1.32	.117
Guided Constructive with Hints	21.2	17.4	3.72	1.32	.005
Guided Constructive without Hints	17.9	21.4	-3.48	1.32	.009
Unguided Constructive	17.8	18.8	-.96	1.32	.469

To explore the relative efficacy of different methods of learning subgoals, a simple main effects comparison was used. The method of learning subgoals affected performance for groups that received feedback, $F(4, 240) = 4.77$, $MSE = 21.9$, $p = .001$, partial $\eta^2 = .073$, and groups that did not receive feedback, $F(4, 240) = 4.72$, $MSE = 21.9$, $p = .001$, partial $\eta^2 = .074$. Based on pairwise comparisons, for participants who did not receive correct response feedback, those in the guided constructive with hints condition performed statistically better than those in the passive condition, Mean Difference = 5.72, $p < .001$. Furthermore, for participants who received feedback, participants in the guided constructive without hints condition performed statistically better than those in the passive and active conditions, Mean Difference = 4.16, $p = .019$; Mean Difference = 5.36, $p = .001$ (see Figure 8). These results suggest that, within both the correct response feedback and no feedback groups, the best performing conditions scored statistically better than those in the worst performing conditions. The other conditions that scored in the middle, such as both unguided constructive groups, were not statistically better or worse than the best or worst performing conditions.

This pattern of results matched the expected pattern of results well, providing support for the hypothesis that there is an optimal combination of support for learning subgoals. In

particular, the disordinal effect of correct response feedback on the guided constructive groups suggests that learners perform best with appropriate support and providing additional types of support hindered learning. The correct response feedback (i.e., the subgoal labels created by experimenters) might have been inappropriate for learners who had already received another type of support (i.e., hints) because the exercise of comparing the feedback to their own responses might not have provided them with useful information. When learners had not received hints while constructing explanations, however, the correct response feedback provided helpful support that improved their performance. Based on these results, it was concluded that providing hints for learners constructing subgoal labels and providing feedback on constructed labels are both techniques that can help learners to perform better on later problem solving, but providing both types of support could hurt performance.

The results for the other groups align with Chi's (2009) framework that passive and active methods of learning produce worse results than constructive learning. The constructive conditions performed numerically better than the passive and active groups though only those that received one type of support performed statistically better. Chi's self-explanation work suggests that working memory is a predictor of success for self-explanations tasks (Wylie & Chi, 2014). One possibility is that for some students a low level of support would be ideal, whereas it would be too little support for others and vice versa. To address this question, individual differences in working memory and quality of subgoal labels were explored.

An ANCOVA was used to explore whether working memory capacity was a covariate of the interventions' effect on problem solving performance. No evidence was found to suggest that working memory capacity affected problem-solving performance, $F(1, 240) = .56$, $MSE = 22.3$, $p = .456$, partial $\eta^2 = .003$. Therefore, it is unlikely that the interaction of method of learning

subgoals and feedback on problem solving performance depended on individual differences in working memory.

Qualitative analysis of subgoal label quality

To determine the quality of participant-created labels, the labels that participants wrote on the worked example during the instructional period were qualitatively analyzed. Each label was analyzed as one unit (i.e., each word within a label was not analyzed individually), and each participant was categorized based on all the labels that they constructed collectively. In nearly all cases, all the labels that a participant created fell into one of the following categories. The coding scheme that was determined *a priori* included categories for whether labels were problem-specific, problem-independent, or incorrect. In all cases except two, participant labels were either completely problem-specific or completely problem-independent.

Problem-specific labels included information about the specific instantiation of the subgoal and, therefore, could be applied only to that one instantiation. For example, the participant-created label "name and add picture to image sprite" could be applied only to the steps that named and added a picture to an Image Sprite. For a participant to be classified as problem-specific, at least 80% of labels had to include information about the details of the problem. The cutoff of 80% was chosen *a priori* to represent four out of five subgoals so that if participants created a problem-independent subgoal label for one of the subgoals, their labels would still be classified as predominately problem-specific.

Problem-independent labels, on the other hand, did not contain any information about the specific instantiation of that subgoal. The subgoal label training specified that labels should be problem-independent. For example, the participant-created label "add properties to app" is problem-independent because it can be applied to any property, such as the name and picture of

an Image Sprite, that is being added to the app. For a participant to be classified as problem-independent, at least 80% of labels had to not include information about the details of the problem. For the same *a priori* reason, if a participant included problem-specific details in labels for one of the subgoals, their labels should still be classified as predominately problem-independent. Problem-independent labels were considered to be higher quality than problem-specific labels because they indicated a more conceptual understanding of the procedure that is more easily applied to solving new problems. Because problem-specific labels include information about the details of the current problem, they cannot be applied directly to novel problems.

Incorrect subgoal labels were those that were execution-based instead of function-based, such as "click on menu," or those that did not describe the correct function (see Table 1). For a participant to be classified as incorrect, more than one label had to meet either of these criteria. In all cases except one, for participants who made incorrect labels, at least 80% of their labels were incorrect.

While implementing this coding scheme, two more categories were defined *post hoc*. For the guided constructive with hints conditions, many of the constructed labels included terms from the hints. For example, the hint for the subgoal that defines the output of an interaction included the term "output," and many participants who received hints included the term "output" in the labels that they created. In all cases, participant-created labels that used terms from the hints were problem-independent. To distinguish these labels from the other problem-independent labels, these labels were classified as hint-term problem-independent labels. For a participant to be classified as hint-term problem-independent, at least three out of the five labels had to include terms from the hints. The cutoff of 60% was chosen *post hoc* in this case to best represent the

data. In all but a few cases, participants either did not use hint terms to construct their labels, or they used hint terms in most of their labels. Therefore, if they used the hint terms in the majority of subgoal labels their use of hint terms was captured by classifying the labels as hint-term problem-independent. If fewer than three labels included terms from the hints, then the participants were classified as problem-independent.

For the unguided constructive conditions, many of the subgoals that participants identified for themselves included many more steps than the subgoals created by experimenters. For example, some subgoals that participants grouped were more than 20 steps long, whereas the longest experimenter-grouped subgoal was seven steps. In all cases, the participant-created labels for these higher level subgoals were problem-specific. For example, one participant identified a subgoal that was 24 steps long and labeled it “make the correct sounds play according to whatever input is received.” To distinguish these labels from the other problem-specific labels, these labels were classified as higher-level problem-specific labels. For a participant to be classified as higher-level problem-specific, the participant-identified subgoals had to include at least twice as many steps the subgoals identified by experimenters because in these cases, participants were lumping two or more experimenter-identified subgoals together. The higher-level problem-specific labels were considered lower quality subgoal labels than the problem-independent or problem-specific labels. One of the benefits of learning the subgoals of a procedure is that subgoals break up long procedures into functional pieces that are easier to adapt to novel problems. The higher-level subgoals did not identify these functional pieces but instead described the procedure that was being executed. Describing the procedure in this way instead of in a functional way is theoretically less conducive to transfer to novel problems.

Two raters scored 20% of participants and compared their scores. Interrater reliability was measured with intra-class correlation coefficient of agreement because the scale of measurement for categories was nominal and absolute agreement was necessary. Reliability was high, ICC(A) = .98, and the remaining 80% of participants were scored by a single rater.

Hints improved participant-created subgoal label quality

For representative examples of participant-created labels for all five classifications, see Table 1. The majority of problem-independent subgoal labels (91%) were two to five words long. This length was similar to the experimenter-created labels. The problem-specific labels, on the other hand, tended to be longer – 54% were longer than five words – because they included problem-specific words, such as specifically mentioning the drum sound.

Participants in the guided constructive with hints conditions created mostly hint-term problem-independent labels or problem-independent labels (62%, see Table 4). About a third of participants in these groups constructed problem-specific labels, and few had incorrect labels. Participants in the guided constructive without hints conditions created worse labels than those who received hints. Again about a third created problem-specific labels, but less than half constructed problem-independent labels. Instead, 19% of participants created incorrect labels, which were not function-based and largely included problem-specific details (e.g., explaining how to complete a step rather than the function of a step).The majority of participants in the unguided constructive conditions created higher-level problem-specific labels. Only a small number of these participants created problem-independent labels, problem-specific labels, or incorrect labels (see Table 4).

Table 4

Percentage of Participants Who Created Problem-Independent, Problem-Specific, or Incorrect Subgoal Labels in the Constructive Learning Conditions.

	Constructive Learning Condition		
	Guided constructive with hints	Guided constructive without hints	Unguided constructive
Problem-independent (% that were hint-term for those with hints)	62% (26%)	45%	7%
Problem-specific (% higher-level for unguided condition)	32%	36%	89% (80%)
Incorrect	6%	19%	4%

To determine whether the type of subgoal labels that participants made affected problem-solving performance, a Kruskal-Wallis *H* test was used. The *H* test was deemed more appropriate than the *F* test for this analysis because the number of participants in each group (i.e., type of subgoal labels) was not equal, violating one of the assumptions of the *F* test. The type of subgoal labels that participants created was also a quasi-experimental variable, making a non-parametric test more valid. The limitation of the *H* test, however, is that it is more conservative than the *F* test. The *H* test was not statistically significant, $p = .12$, though the median scores (reported here instead of means because the *H* test uses median scores) were numerically higher for problem-independent hint-term labels (median for problem-independent hint-term = 23 out of 25) than for all other groups (median for problem-independent = 19, median for higher-level problem-specific = 19, median for problem-specific = 19.5). The average standard deviation for these groups was 4.95, making the error too large to find statistically significant differences between groups. Based on these data, there was not sufficient evidence to suggest that participants who created problem-independent hint term labels performed better than those who created other types of labels, but this area warrants more exploration.

Though the types of labels that participants created were not found to directly affect problem solving performance, most of the participants in the guided constructive with hints conditions created subgoal labels that described similar function as the experimenter-created

labels, meaning that they created labels that aligned with those created through an intensive task analysis with a subject-matter expert. For this reason, these participant-created labels were considered high quality subgoal labels. Participants creating high quality labels might explain why participants performed better on the problem-solving tasks when they did not receive correct response feedback (i.e., experimenter-created labels) compared to when they did receive feedback. For participants who created high quality labels, comparing their labels to the experimenter-created labels might not have been as beneficial as reviewing the labels that they constructed, as participants in the no feedback condition did. Comparing labels might have caused participants to unjustifiably question or doubt their understanding of the procedure, whereas reviewing their own labels would reinforce the mental representations that participants developed. This effect is similar to the expertise-reversal effect in which giving instructional support to students helps their learning if they have a low level of prior knowledge but hinders their learning if they have a high level of prior knowledge (Sweller, 2010).

Participants in the guided constructive without hints conditions made more problem-specific or incorrect labels (55%) than those who received hints (38%). Therefore, on average these participants had lower quality labels than those who received hints. This difference might explain why participants who did not receive hints performed better when they received correct response feedback than when they did not. The feedback likely provided necessary support for these participants to decontextualize their knowledge of the procedure, improving their problem-solving performance.

Most participants in the unguided constructive conditions grouped subgoals that were different than the subgoals identified by the experimenter and labeled these subgoals with problem-specific labels. Because these labels were different from the experimenter-created labels

in multiple aspects, it is not surprising that pre-canned, correct response feedback did not affect performance for the unguided constructive conditions. The feedback likely provided guidance that was so different from the participants' mental representations of the procedure that they could not reconcile the two different representations.

Participants in the unguided constructive condition also spent much more time looking at the feedback than those in other conditions. A main effect of subgoal learning method was found for time spent looking at feedback, $F(4, 240) = 9.84$, $MSE = 2.21$, $p < .001$, partial $\eta^2 = .15$ (see Figure 9). Using Tukey's HSD post-hoc analysis, the unguided constructive group was the only group found to have a statistically significant mean difference from the passive group (Mean Difference = 1.40, $p = .017$), active group (Mean Difference = 1.64, $p < .001$), guided constructive with hints (Mean Difference = 1.45, $p < .001$), and guided constructive without hints (Mean Difference = 1.43, $p < .001$). There was no main effect of feedback condition on feedback time, $F(1, 240) = .46$, $MSE = 2.21$, $p = .50$, partial $\eta^2 < .01$, or interaction of subgoal learning method and feedback, $F(4, 240) = .79$, $MSE = 2.21$, $p = .50$, partial $\eta^2 = .01$. These time on task results further suggest that participants in the unguided constructive conditions had difficulty reconciling the labels that they created with those presented in the correct response feedback, making the experimenter-created labels a poor source of feedback for this group.

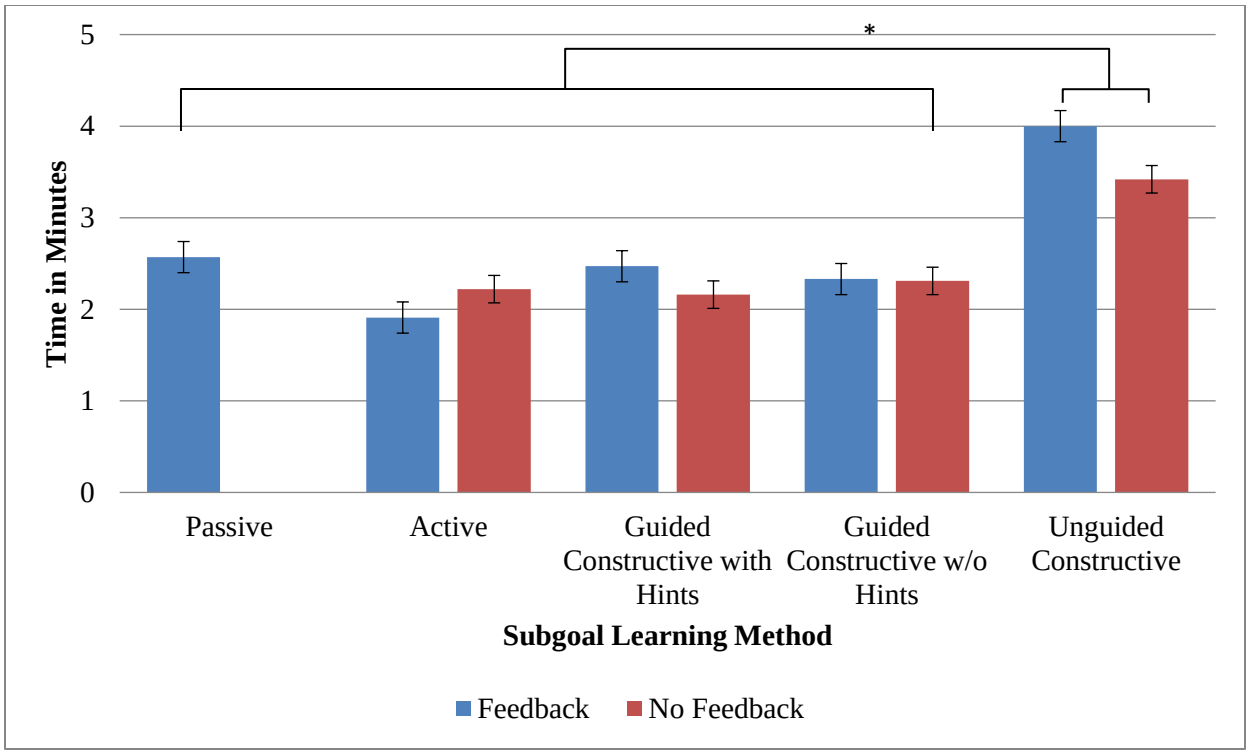

Figure 9. Time spent on reviewing feedback among conditions. Error bars are standard error. Statistically significant differences are indicated with asterisks.

Guided constructive methods of learning took longest

Time that participants spent on each part of the experimental session was collected. There were differences among groups for time spent on the worked example and time spent working on practice problems. For time spent on the worked example, which included using the worked example to re-create the app and learning the subgoals of the procedure through passive, active, or constructive methods, there was a main effect of subgoal learning method, $F(4, 240) = 25.00$, $MSE = 32.80$, $p < .001$, partial $\eta^2 = .29$ (see Figure 10). The passive (M = 25.5 minutes, SD = 5.9) and unguided constructive (M = 27.1 minutes, SD = 6.5) groups completed this part of the instructional period quickest and were not statistically different from each other (Mean Difference = 1.58, $p = .64$). The active group (M = 31.1 minutes, SD = 5.4) took statistically significantly longer than the passive group (Mean Difference = 5.63, $p < .001$) and the unguided

constructive group (Mean Difference = 4.04, p = .004). The guided constructive groups took statistically longer than the active group (with hints, M = 34.7, SD = 6.05, Mean Difference = 3.54, p = .019; without hints, M = 33.85, SD = 5.39, Mean Difference = 3.69, p = .041) and were not statistically different from each other (Mean Difference = 0.82, p = .95).

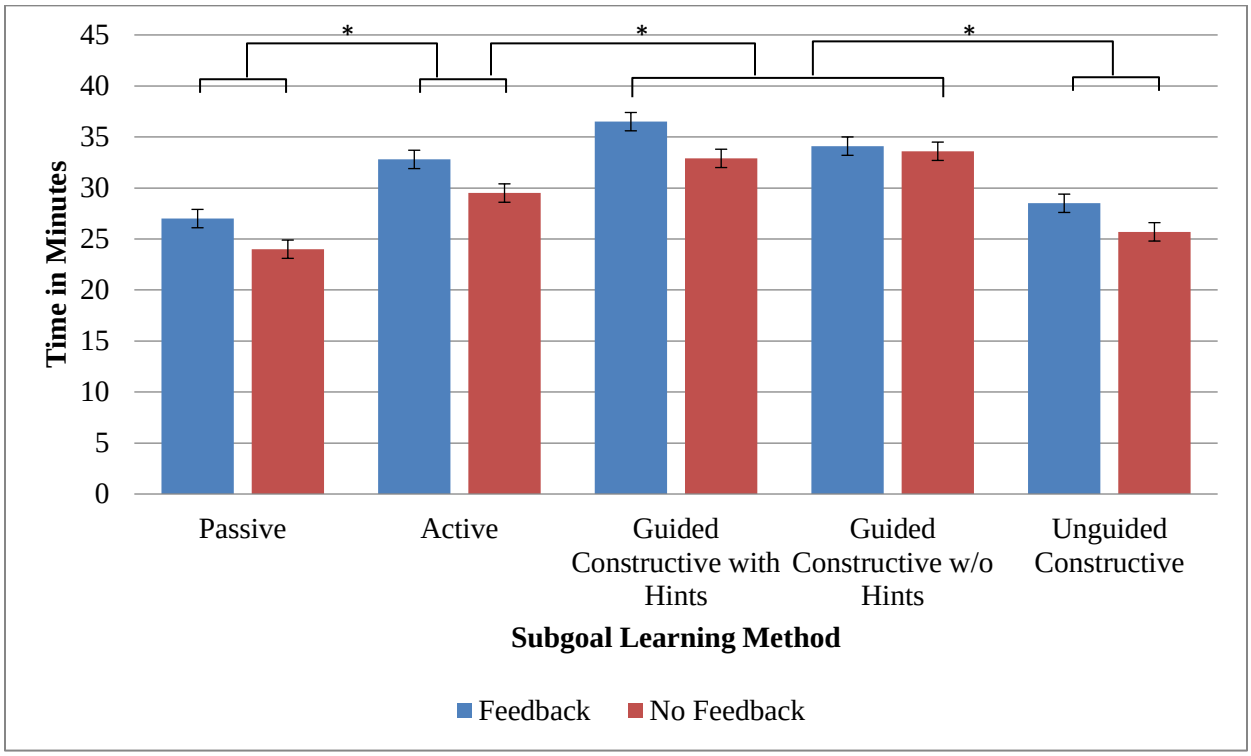

Figure 10. Time spent using the worked example, including re-creating the app and engaging in subgoal learning methods, among conditions. Error bars are standard error. Statistically significant differences are indicated with asterisks.

Except for the unguided constructive group, the constructive methods of learning subgoals took longer to complete than the non-constructive methods. These results were expected because constructing knowledge takes more thought and, therefore, time to complete. The unguided constructive group might have taken less time because participants tended to construct high-level subgoal labels that described the process of creating the Music Maker app instead of the conceptual procedure for creating apps. This level of description is much easier to

identify than a deeper, conceptual description. There was no main effect of feedback on time spent using the worked example, $F(1, 240) = 1.41$, $MSE = 32.8$, $p = .21$, partial $\eta^2 = .03$, or interaction of subgoal learning method and feedback, $F(4, 240) = .58$, $MSE = 32.8$, $p = .67$, partial $\eta^2 = .01$. Whether participants received feedback did not affect the time they spent using the worked examples. This result was expected because this measurement was taken before participants knew that they would receive feedback.

Correct response feedback increased time spent on practice problems but not on assessments

For time spent on practice problems, a main effect of feedback was found, $F(1, 240) = 6.14$, $MSE = 9.4$, $p = .014$, partial $\eta^2 = .025$ (see Figure 11). Participants who received correct response feedback ($M = 9.92$ minutes, $SD = 3.16$) spent an extra 11% of time on solving practice problems than those who did not ($M = 8.96$ minutes, $SD = 3.0$). This effect might be due to participants referencing both the worked example and correct response feedback while solving practice problems instead of referencing only the worked example. The effect accounts for only 2.5% of the variance in time, however, so the effect of feedback on practice problem time is small.

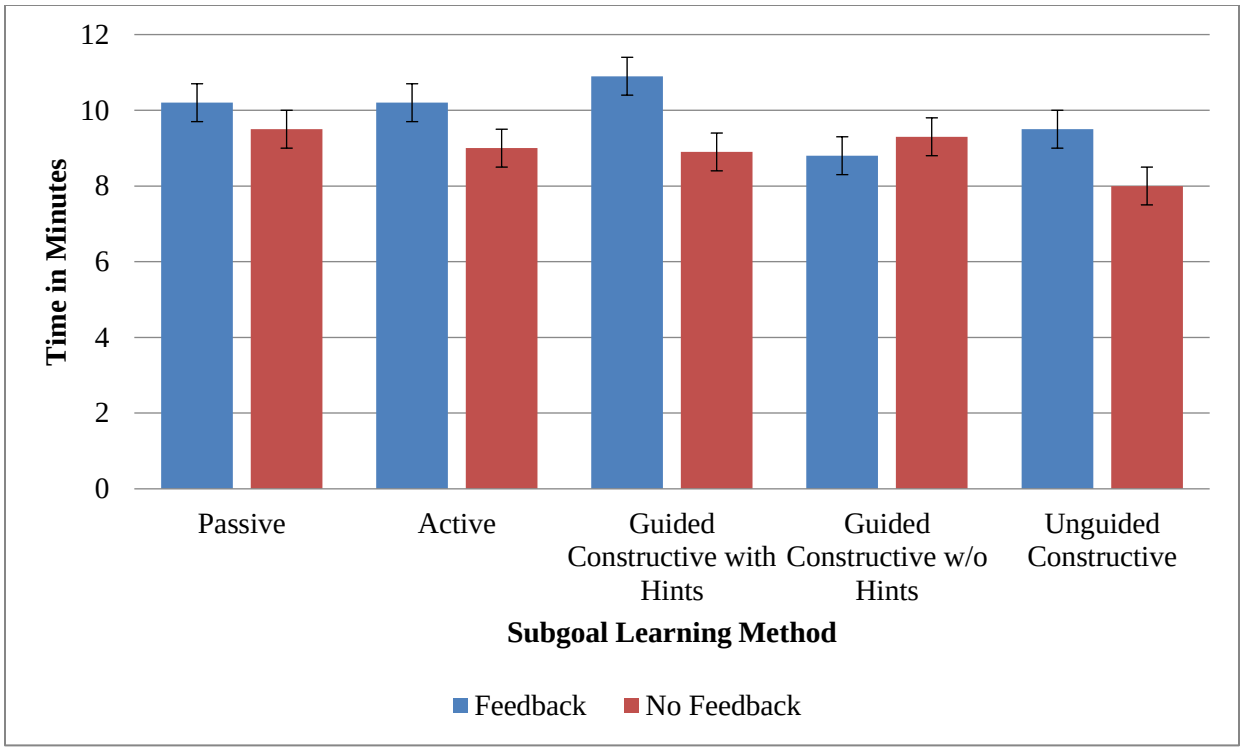

Figure 11. Time spent working on practice problems among conditions. Error bars are standard error.

There was no main effect of subgoal learning method on time spent working on practice problems, $F(4, 240) = 1.35$, $MSE = 9.4$, $p = .25$, partial $\eta^2 = .02$, or interaction of subgoal learning method and feedback, $F(4, 240) = 1.25$, $MSE = 9.4$, $p = .29$, partial $\eta^2 = .02$. Method of subgoal learning, therefore, did not affect the time participants spent working on practice problems.

The last time measurement was time spent on problem-solving tasks. Participants spent an average of 23.52 minutes on the problem-solving tasks ($SD = 2.83$). No differences among conditions were found for this measurement. There was no main effect of subgoal learning method, $F(4, 240) = 1.51$, $MSE = 7.8$, $p = .15$, partial $\eta^2 = .027$, no main effect of feedback, $F(1, 240) = 3.55$, $MSE = 7.8$, $p = .06$, partial $\eta^2 = .015$, and no interaction of method and feedback, $F(4, 240) = 1.15$, $MSE = 7.8$, $p = .33$, partial $\eta^2 = .02$. Based on these results, the interventions

did not affect the time it took participants to complete the problem-solving tasks. Groups that performed better or worse on problem solving performance did not differ on the amount of time that it took to solve problems.

No differences found in other metrics

The purpose of the explanation assessment was to test participants' knowledge of the problem-solving procedure independent from their problem-solving performance. For the explanation assessment, participants received a point for each step that was correctly paired with its functional label. The maximum possible score was 20. The mean score on this assessment for all groups was 15.8 with a standard deviation of 4.20. No statistical differences were found for performance on the explanation task among the conditions. There was no main effect of subgoal learning method, $F(4, 240) = 1.27$, $MSE = 17.52$, $p = .28$, partial $\eta^2 = .02$, no main effect of feedback, $F(1, 240) = .17$, $MSE = 17.52$, $p = .68$, partial $\eta^2 = .001$, and no interaction, $F(4, 240) = 1.66$, $MSE = 17.52$, $p = .16$, partial $\eta^2 = .02$. These results suggest that participants in all conditions were equally prepared to complete the explanation task, regardless of whether they had seen the experimenter-created labels in the instructions or not. Because participants could match the functions of subgoals to the experimenter-created labels even if they had not seen the labels before, this finding suggests that participants could equally recognize the correct experimenter-created label that matched subgoals' functions. All participants, therefore, could recognize the subgoals of the function, but only two of the guided constructive conditions performed better on problem solving, suggesting that participants in those conditions could better apply their knowledge.

At the end of the instructional period, including worked example, feedback or review, and practice problems, participants were asked to rate their cognitive load while learning the

procedure. This measure was intended to assess whether there were significant cognitive load differences between the conditions that might affect participants' experience of learning. Overall self-report of cognitive load was not affected by the method of subgoal learning, $F(4, 240) = 1.44$, $MSE = 156.8$, $p = .21$, presence of feedback, $F(1, 240) = .70$, $MSE = 156.8$, $p = .40$, or their interaction, $F(4, 240) = 1.34$, $MSE = 156.8$, $p = .26$. On average, participants rated all types of cognitive load in the middle (i.e., 35%-45% on average, see Figure 12). In addition, no differences were found within each of the three types of cognitive load: intrinsic, extraneous, and germane (see Table 5).

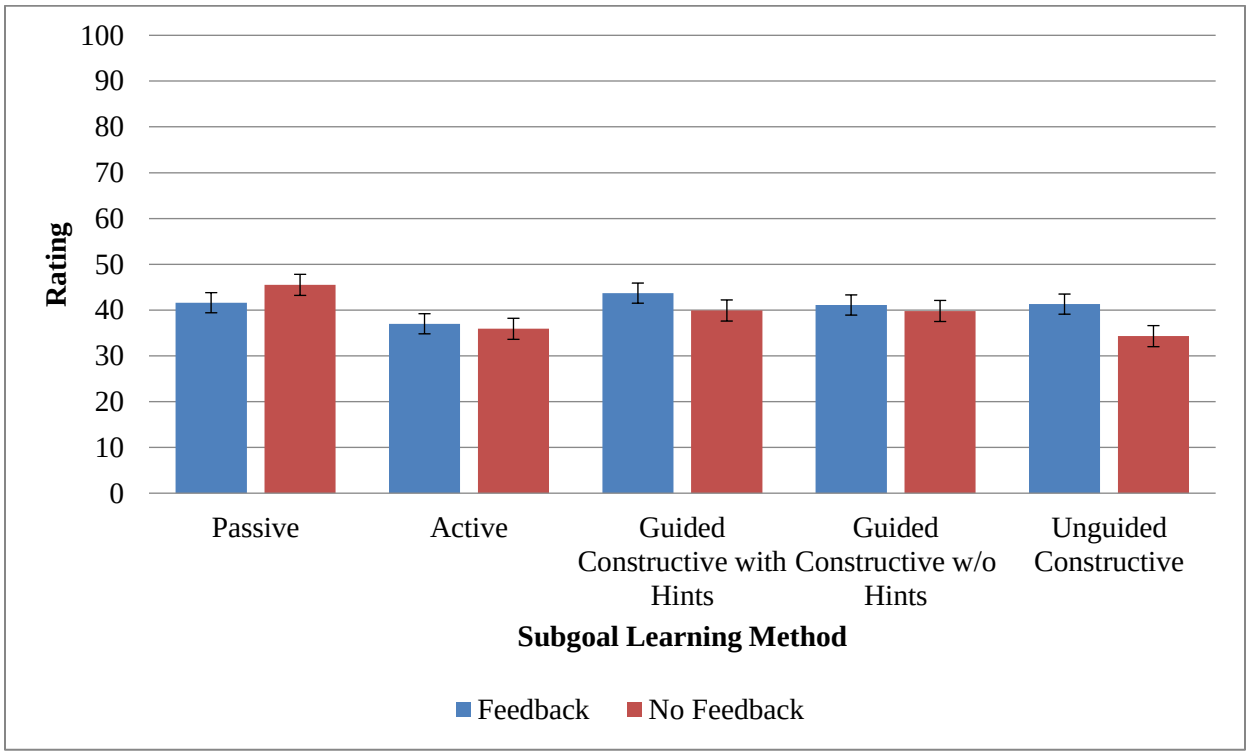

Figure 12. Self-reported rating of cognitive load while working through the instructional period. Error bars are standard error.

Table 5

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

	Main Effect of Subgoal Learning Method		Main Effect of Feedback		Interaction	
	F	*p*	*F*	*p*	*F*	*p*
Intrinsic Load	1.03	.39	.18	.67	.52	.72
Extraneous Load	.18	.95	.05	.83	.77	.55
Germane Load	1.99	.10	.72	.40	1.42	.23

These results suggest that participants did not perceive differences in cognitive load among the conditions; therefore, the participants constructing labels did not perceive a higher cognitive load than participants performing more guided tasks. It is important to note along with this finding that the less guidance that constructive participants had, the worse their constructed subgoal labels were. The guided constructive without hints condition had fewer problem-independent labels than the with hints condition (though they still solved problems well given correct response feedback), and the unguided constructive condition created mostly higher-level, problem-specific labels. It is possible, therefore, that participants could have created better labels if the task had demanded it. For example, if students who received unguided worked examples were told that their subgoals must include no more than five steps and that they had to repeat their labels multiple times in the example, then they would likely have created better labels but also experience higher cognitive load. The balance between cognitive load and performance is important to consider because if constructing high-quality labels is too cognitively taxing, learners might be less inclined to do it, even if it improves learning.

Participants were asked to rate how well they understood the instructions from "1 – Not well at all" to "7 – Very well." In general, participants rated that they understood the instructions well (M = 5.96, SD = 1.06). These ratings were not affected by the method of subgoal learning, $F(4, 240) = .87$, $MSE = 1.13$, $p = .48$, presence of feedback, $F(1, 240) = .36$, $MSE = 1.13$, $p =$

.55, or their interaction, $F(4, 240) = .81$, $MSE = 1.13$, $p = .52$. Participants were also asked to rate how comfortable they were solving novel problems from "1 – Not comfortable at all" to "7 – Very comfortable." Participants rated that they were comfortable solving new problems ($M = 5.60$, SD = 1.19). These ratings were not predicted by the method of subgoal learning, $F(4, 240) = 1.99$, $MSE = 1.41$, $p = .10$, presence of feedback, $F(1, 240) = .32$, $MSE = 1.41$, $p = .57$, or their interaction, $F(4, 240) = 1.13$, $MSE = 1.41$, $p = .34$. These results indicate that participants in different conditions felt equally prepared to solve novel problems, even though some of them performed better than others. Because perceived understanding and comfort solving novel problems were equivalent across groups in this study, these factors were not expected to have affected participants' problem-solving performance.

In summary, the experiment explored the tradeoffs between instructional guidance and constructing knowledge for learning a procedure. The results suggested that constructive methods of learning subgoals were the most effective, but they required some instructional support. Either receiving correct response feedback on constructed labels or receiving hints while constructing labels, but not both, led to the best problem-solving performance. Participants who received hints while constructing labels were more likely to construct problem-independent labels that are readily applicable to a range of problems than participants who did not receive hints. These participants performed better when they did not receive correct response feedback than when they did, suggesting that, for those who received hints, the feedback was not an appropriate instructional support to promote constructive learning. In contrast, participants who did not receive hints performed better when they received correct response feedback than when they did not, suggesting that the feedback was necessary for the best performance when participants did not receive hints, perhaps because it helped them to recognize the problem-

independent functions of the procedure. Correct response feedback did not improve performance for participants in the unguided constructive condition. Because participants tended to divide the worked example into subgoals that were different than those identified in the correct response feedback, they likely could not easily use the feedback to compare to the labels that they had created, making them ineffective.

Conclusions

Subgoal learning has been primarily supported through passive methods: subgoal labeled instructions. These methods have been successful at improving problem solving performance in procedural domains because they give learners beneficial instructional guidance (e.g., Catrambone, 1998). Passive methods, however, are typically less effective for learning than active and constructive learning methods (Chi, 2009). The primary goal of the present study was to further improve problem solving performance by exploring active and constructive methods of learning subgoals. The results suggest that guided constructive methods of learning subgoals by self-explaining the subgoals of a well-structured problem can lead to better problem-solving performance compared to passive, active, and unguided constructive methods. This finding means that learners can benefit from instruction that guides them to self-explain what instructions would typically directly explain. Guided constructive methods of learning subgoals were most effective when the instructions either provided hints while learners were self-explaining the worked example by creating labels or correct response feedback after they created labels, but not when the instructions provided both. This finding supports the idea that combining different types of guidance inappropriately can hinder learning.

The present experiment taught college-level, novices to program using Android App Inventor. The results, therefore, suggest that constructive learning can be better than passive

learning, even for a complex problem-solving procedure, such as programming (Morrison, 2013), and even for novices. For a task that has a different level of complexity or for learners at a different level of knowledge, the results might have turned out differently. More complex tasks generally require more instructional support to adequately guide novices. For example, if the task was more complex, learners might have needed more support to self-explain and learn subgoals constructively. In contrast, if the task was less complex, learners might not have needed as much support to understand the procedure well and might benefit from having less instructional support and more opportunities to construct knowledge. Similarly, for learners with more knowledge, providing less instructional support is typically associated with better learning because students have more opportunities to self-explain and construct knowledge for themselves.

The pattern of results suggests that the role of feedback in constructive learning should be carefully considered. In this experiment, correct response feedback was found to hinder problem solving performance when learners had also received guidance while creating subgoal labels. This finding is particularly important for many educational technologies that are being developed to provide correct response feedback to students. Although it is not unprecedented to find that correct response feedback hinders constructive learning, usually the cause is attributed to learners' overreliance on feedback as a form of instructional support (e.g., Schworm & Renkl, 2006). In the present study, however, learners were not aware that they would receive feedback until after they finished studying the instructions, meaning that they could not rely on the information provided through feedback. Still the results show that when learners received hints during the constructive learning activity, receiving correct response feedback hindered their later problem-solving performance. This finding provides evidence that feedback that is not

responsive to learners' construction of knowledge could diminish learning outcomes rather than improve them.

In this experiment as well, feedback improved problem-solving performance when learners constructed subgoal labels with guidance (i.e., the worked example was already divided into subgoals) but without hints. The study found no differences in problem solving performance between learners who received hints during the constructive learning activity or those who received correct response feedback after the constructive learning activity. Therefore, there is no evidence that one type of instructional support is better than the other for learning. The quality of subgoal labels created by participants, however, was better when learners received hints than when they did not. This difference in subgoal label quality was not related to performance on any of the metrics in the present study, but it does suggest that participants had a better mental organization of information related to the procedure at this point in time. It is tenable that future work could find that higher quality labels are related to better retention or performance on related problem-solving procedures. It is also tenable that the correct response feedback improved learners' mental organizations and no meaningful differences among the learners persisted after the feedback was given.

Limitations and Future Work

Feedback might have hindered learning in this case because it required learners who had created good, problem-independent self-explanations to compare their explanations with that of the experimenter. The reasons that this comparison could be detrimental were not explored in this research, which greatly narrows the generalizability of this finding. Despite this limitation of the current work, possible explanations will be discussed here. Even if the explanations created by the participants and those created by the experimenter were similar, participants might not

have had enough domain knowledge to recognize how similar the explanations were. For a participant who created good, problem-independent subgoal labels, comparing the two explanations could have had two negative effects: cause confusion and unnecessarily high cognitive load in the learner who is unable to reconcile the explanations that they created and those that the experimenter created and/or cause the learner to abandon their explanations and use what they might have perceived to be the only correct explanations. Both effects would negate the benefits of constructive learning – building knowledge upon prior knowledge in an organization that makes sense to the learner.

To explore whether the comparison between good participant-created and experimenter-created explanations is the cause of feedback's negative effect when learners received the guided constructive example with hints, a yoked experimental design could be employed. The goal of this design would be to use students' terms (either their own or that of their yoked partner) in the correct response feedback, which in this case would be responsive to participants' work, to determine whether differences in terms between the student explanations and the feedback explain the decrease in performance for the guided constructive with hints group. In this design, participants could be given the guided constructive with hints condition and asked to create their own subgoal labels. Then participants would be grouped into yoked pairs and receive either feedback based on the labels that they had created (i.e., personalized correct response feedback) or feedback based on the labels that their yoked partner had received (i.e., not personalized feedback but also not canned knowledge of correct response feedback). For example, if the non-yoked participant created a correct label, such as "Add component to app," then the feedback would use the same language that the participant had used, such as "Add component." The yoked participant would see this feedback too, regardless of the label that they had created. Similar to

canned feedback, if the yoked participant had also used the term component, then the feedback would align well. If the yoked participant has used a different term, however, then the feedback would not align well, making the yoked condition a good control group. If the non-yoked participant created an incorrect label, such as "Add ImageSprite," then the feedback would use similar, but corrected, language, such as "Add component." Again, the yoked participant would see the same feedback. The feedback based on participant-created labels would be advertised to both groups as correct labels developed by an expert.

It is hypothesized that participants who received personalized feedback based on their labels would not have difficulty integrating their created labels with the feedback labels; therefore, it is hypothesized that this group would perform similarly to those who did not receive feedback. If some participants in this group created problem-specific labels, the feedback would provide a problem-independent version of their label. If they created incorrect labels, then the feedback would default to the experimenter-created labels. In both cases, these participants might perform better on novel problem-solving tasks because their problem-specific or incorrect labels are corrected to problem-independent labels. It seems unlikely that providing feedback for learners who created good subgoal labels would further improve problem solving performance unless the learners were uncertain of their labels and would benefit from validation of their labels.

For participants who receive yoked feedback, it is hypothesized that they would perform as poorly as or worse than participants in the present experiment who received experimenter-created labels as feedback. These participants would receive feedback labels that would be different enough from their own, unless they used the same words as their partner, that it is expected that the participants would have trouble reconciling the two sets of labels. These

participants might even perform worse because the feedback labels would be created by another participant who is a novice in the subject matter and in making subgoal labels; therefore, it is possible that the yoked feedback would make even less sense than experimenter-created labels to the participants in the yoked condition.

Future work could also focus on exploring whether constructive methods of subgoal learning could be developed into a general learning strategy. Perhaps teaching learners to create their own subgoal labels would help them to improve performance on a range of tasks. After constructing, with adequate support, subgoal labels for several procedures, learners might become skilled at developing labels and would be able to construct labels for new procedures in different domains without hints or feedback. Eventually, they might even be able to breakdown problem solving procedures into subgoals by themselves and benefit from subgoal learning without help from instructors or instructional designers. This strategy would likely have benefits that are similar to training students to self-explain in procedural domains. Transferring learning strategies, however, from one domain to another, especially without any guidance, is typically difficult to achieve (Brown, 1992); therefore, more work would be necessary to explore this possibility. If it is possible, training learners on this type of learning strategy could help learners perform better--both at initial learning and later transfer--across a range of procedural fields.

The present study suggests that learners are better able to solve novel problems when they learn the subgoals of a well-structured procedure through constructive methods that provide an appropriate combination of types of guidance than when they learn subgoals through passive or active methods. It is critical to note that this research was conducted in an independent learning environment in which learners did not have access to an instructor or peers and, therefore, could not discuss their self-explanations with others. Much of the recent research on

constructive learning is conducted in classrooms or otherwise social environments in which learners can exchange ideas and solicit instructor feedback. The personalized feedback provided in these cases is dynamically responsive to the learners' questions, statements, and non-verbal cues, meaning that learners do not have to take on the burden of comparing their explanations with an expert's explanation and self-monitoring whether they have fully reconciled both explanations. Learners in the present study did not have those resources, but some of the participants assigned to the constructive learning method still performed better than those assigned to the passive learning method. This suggests that even without access to personalized feedback, which is almost always preferable but almost always more expensive to deliver, constructive learning can be more effective than passive learning.

Because the interventions in the present study are not reliant on a social learning environment, they would be relatively easy to implement in a range of instructional environments, including technology-supported environments. Chunking problems into subgoals and providing hints that help learners to realize the similarities between different instances of subgoals would be an easy intervention to include in instructional material because it does not need to be customized for each individual learner. If providing hints helps students learn constructively as much as providing correct response feedback, as this study suggests, then constructive learning can be supported in a larger range of learning environments. By receiving hints, learners can constructively learn subgoals in learning environments that do not provide feedback, or at least not immediate feedback, like many online learning environments. If future work suggests that constructing subgoal labels can be a general learning strategy applied to procedures in various domains, then the learning methods in the present research will become even more compelling. Based on the findings of the present study, the best subgoal learning

outcomes should be achieved through constructive methods with appropriate guidance, which does not mean a combination of all available types of guidance.

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

Berthold, K., Eysink, T. H. S., & Renkl, A. (2009). Assisting self-explanation prompts are more effective than open prompts when learning with multiple representations. *Instructional Science, 37,* 345-363. doi:10.1007/s11251-008-9051-z

Bielaczyc, K., Pirolli, P. L., & Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem solving. *Cognition and Instruction, 13*(2), 221-252. doi:10.1207/s1532690xci1302_3

Bjork, R. A. (1994). Memory and metamemory considerations in the training of human beings. J. Metcalfe & A. P. Shimamura (Eds.). *Metacognition: Knowing about Knowing*. MIT Press: Cambridge, MA.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.) (2000). *How people learn: Brain, mind, experience, and school: Expanded edition.* Retrieved from http://www.nap.edu/catalog.php?record_id=9853

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, *2*(2), 141-178.

Catrambone, R. (1994). Improving examples to improve transfer to novel problems. *Memory and Cognition, 22,* 605-615. doi:10.3758/BF03198399

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). Task analysis by problem solving (TAPS): Uncovering expert knowledge to develop high-quality instructional materials and training. Paper presented at the 2011 Learning and Technology Symposium (Columbus, GA, June).

Catrambone, R. & Holyoak, K. J. (1989). Overcoming contextual limitations on problem-solving transfer. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15*(6), 1147-1156.

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

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, 145-182.

Chi, M. T. H., de Leeuw, N., Chiu, M., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18*(3), 439-477.

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.

de Koning, B. B., Tabbers, H. K., Rikers, R. M. J. P., & Paas, F. (2011). Improved effectiveness of cueing by self-explanations when learning from a complex animation. *Applied Cognitive Psychology, 25*, 183-194.

deWinstanley, P. A., Bjork, E. L., & Bjork, R. A. (1996). Generation effects and the lack thereof: The role of transfer-appropriate processing. *Memory, 4*, 31-48.

Eiriksdottir, E., & Catrambone, R. (2011). Procedural instructions, principles, and examples: How to structure instructions for procedural tasks to enhance performance, learning, and transfer. *Human Factors, 53*(6), 749-770. doi:10.1177/0018720811419154

Graesser, A. C., Hu, X., Nye, B., & Sottilare, R. (2016). Intelligent tutoring systems, serious games, and the Generalized Intelligent Framework for Tutoring (GIFT). *Using games and simulation for teaching and assessment*, 58-79.

Hausmann, R. G. M., & Chi, M. T. H. (2002). Can a computer interface support self-explaining? *Cognitive Technology*, *7*(1), 4-14.

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

Jacoby, L. L. (1978). On interpreting the effects of repetition: Solving a problem versus remembering a solution. *Journal of Verbal Learning & Verbal Behavior, 17*, 649-667.

Kirschner, P., Sweller, J., & Clark, R. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75-86.

Kulik, J. A., & Fletcher, J. D. (2016). Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of Educational Research, 86*(1), 42-78.

LeFevre, J., & Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction, 3*, l-30. doi:10.1207/s1532690xci0301_1

Linn, M., & Clancy, M. (1992). The case for case studies of programming problems. *Communications of the ACM, 35*(3), pp 121-132.

Ma, W., Adesope, O. O., Nesbit, J. C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology, 106*(4), 901-918.

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

Maxwell, S. E., & Delaney, H. D. (2004). *Designing experiments and analyzing data: A model comparison perspective* (2nd ed.). New York, NY: Psychology Press.

Molloy, E. K., & Boud, D. (2014). Feedback models for learning, teaching and performance. In *Handbook of research on educational communications and technology* (pp. 413-424). Springer New York.

Morrison, B. B. (2013, August). Using cognitive load theory to improve the efficiency of learning to program. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 183-184). ACM.

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.

Newman, R. S. (1998). Students' help seeking during problem solving: Influences of personal and contextual achievement goals. *Journal of Educational Psychology*, *90*(4), 644.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.

Polson, M. C., & Richardson, J. J. (Eds.). (2013). *Foundations of Intelligent Tutoring Systems*. Psychology Press.

Renkl, A. (2002). Worked-out examples: Instructional explanations support learning by self-explanations. *Learning and Instruction, 12*, 529-556.

Renkl, A. (2005). The worked-out-example principle in multimedia learning. In R. Mayer (Ed.), Cambridge handbook of multimedia learning. Cambridge, UK: Cambridge University Press.

Renkl, A., & Atkinson, R. K. (2002). Learning from examples: Fostering self-explanations in computer-based learning environments. *Interactive Learning Environments, 10*(2), 105-199. doi:10.1076/ilee.10.2.105.7441

Renkl, A., & Atkinson, R. K. (2003). Structuring the transition from example study to problem solving in cognitive skills acquisition: A cognitive load perspective. *Educational Psychologist, 38*, 15–22.

Renkl, A., Stark, R., Gruber, H., & Mandl, H. (1998). Learning from worked-out examples: The effects of example variability and elicited self-explanations. *Contemporary Educational Psychology, 23*, 90-108.

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In *Computer-supported collaborative learning*, 128, pp. 69-197).

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.

Schworm, S., & Renkl, A. (2006). Computer-supported example-based learning: When instructional explanations reduce self-explanations. *Computers & Education, 46*, 426-445.

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.

Thurlings, M., Vermeulen, M., Bastiaens, T., & Stijnen, S. (2013). Understanding feedback: A learning theory perspective. *Educational Research Review, 9*, 1-15.

Van der Kleij, F. M., Feskens, R. W., & Eggen, T. M. (2015). Effects of feedback in a computer-based learning environment on students' learning outcomes. *Review of Educational Research, 85*(4), 475-511.

Van Gog, T., Paas, F., & Van Merrienboer, J. J. G. (2004). Process-oriented worked examples: improving transfer performance through enhanced understanding. *Instructional Science, 32*(1-2), 83-98.

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.