Identifying Design Principles for CS Teacher Ebooks through Design-Based Research

Barbara Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, Mark Guzdial
Georgia Institute of Technology College of Computing
801 Atlantic Drive
Atlanta, GA 30332
ericson@cc.gatech.edu

ABSTRACT

Several countries are trying to provide access to computing education for all secondary students. However, there are not enough teachers who are prepared to teach computer science. Interactive electronic books (ebooks) are a promising approach for providing low-cost professional development in computer science. Over the last four years, our research group has been conducting design-based research by iteratively developing and testing versions of a teacher ebook to help secondary teachers with no programming experience learn to teach an introductory programming course. The interactive elements in the ebook were designed based on research results from educational psychology and are intended to make learning more efficient and effective. Our goals for this effort are to increase teachers' knowledge of computer science concepts and to improve teachers' confidence in their ability to teach computer science. In this paper we summarize our previous work and report on a large-scale study of version two of the teacher ebook. We also recommend several design principles for interactive ebooks for computing teachers based on feedback from teachers, log file analyses, and randomized controlled studies.

Categories and Subject Descriptors

Social and professional topics ~ Computing education

General Terms

Measurement; Design; Experimentation

Keywords

high school teacher professional development; ebooks; electronic books; design-based research

1. GOAL: MORE CS TEACHERS

The United States president, Barack Obama, announced a new initiative in January, 2016, *Computer Science for All [33]*. President Obama declared that computer science "is a new basic skill." In his 2016 State of the Union address, he said that we should offer every student computer science classes. There are similar initiatives in the United Kingdom [8; 9], New Zealand [4; 5] Denmark [10], and other countries around the globe.

The challenge in all of these efforts is to find enough teachers to staff computer science classrooms. Since computer science is not currently considered a core subject in most states in the US, we do not have accurate national records about the availability of introductory computer science courses. We do know that fewer than 10% of high schools in the United States offer the Advanced Placement Computer Science A course, which is equivalent to a college level CS1 course [13]. In New York City, the largest school system in the United States, less than 10% of the high schools have any computer science teacher at all [39].

An effective computer science high school teacher has content knowledge about computer science as well as pedagogical content knowledge (PCK) about how to teach computer science [30]. Our goal for professional development is to increase teacher self-confidence, in addition to increasing content knowledge and PCK. Ni found that teacher self-confidence is a critical factor in getting teachers to pursue further learning in computer science [28]. We know that CS teacher self-confidence about teaching depends greatly on their confidence that they can do the programming needed in their course [30].

In the United States, a new Advanced Placement course and exam is in development, *CS Principles*. A goal for this course (AP CSP) is to broaden participation in computing [1]. In support of that project, we provide free on-line professional development to teachers learning the programming part of CSP. We chose programming because we knew that it was critical for developing teacher self-confidence. We chose to build an electronic book (*ebook*) to provide that professional development freely at scale.

The most important outcome of our project is the design principles that can inform future ebooks for CS teacher professional development. To develop the design principles, we adopted a *design-based research* (DBR) approach. We started with hypotheses based on learning theories, developed the book based on those hypotheses, tested the ebook with real teachers, analyzed the data from the tests, then modified our hypotheses. We used our modified hypotheses to inform re-development of the book, and iterated the cycle. In parallel, we conducted laboratory experiments to test our hypotheses using randomized controlled studies.

In this paper, we describe our process and the design principles that we have generated from our iterations and experiments. We present our arguments for our ebook approach and describe how the features of our ebook are informed by theory about learning and teacher professional development. This paper summarizes the results from our prior studies and presents the new results from a large-scale study of version 2 of the teacher ebook, which was conducted during the spring and summer of 2015.

2. METHODS

We use a *design-based research* methodology to frame our work. Design-based research is an iterative approach to educational

ICER '16, September 08-12, 2016, Melbourne, VIC, Australia
© 2016 ACM. ISBN 978-1-4503-4449-4/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2960310.2960335

research that grew out of the work of Ann Brown, an educational psychologist, and Alan Collins, a learning scientist [7]. The goal of design-based research is to both improve practice and generate or advance theories that transcend the particular context [3]. We are drawing on theories from educational psychology and trying to advance those theories by testing them in actual learning environments. However, we also conduct randomized controlled experiments to validate the hypotheses we generate.

2.1 Why Ebooks

The general problem we are addressing is how to provide professional development opportunities to working (*in-service*) secondary school teachers to prepare them to teach CS. We started our work by asking two questions: (1) what do expert computer science high school teachers do and (2) how do we teach CS within the time limitations of a working teacher.

In our studies of expert CS teachers [15; 28], we realized that the tasks and skills of a high school CS teacher are unlike those of a professional software developer. We found that CS teachers rarely write code. They often help students debug code, and they teach students how to evaluate their own and peer's code [19]. Expert teachers know what students get wrong, how to diagnose those misconceptions, and what activities help students develop a better understanding [32], i.e., pedagogical content knowledge (PCK) [21].

In an interview study of working professionals taking post-graduate CS classes on-line, we found that many learners did not succeed, and for reasons that had little to do with the course itself [6]. We found that learners dropped out of the course because of events in their lives outside of the class, which made it impossible for them to keep up with the pace of the course. The pedagogy of typical CS classes, with a focus on apprenticeship via programming activities, led to unpredictably long work sessions dealing with mundane issues (e.g., "hours because one comma was out of place"). The working professionals from that study are similar to the working teachers we are studying.

Another scalable way to deliver low-cost professional development to teachers is through Massive Open Online Courses (MOOCs). However, MOOCs have had low completion rates, which is problematic when the goal is to prepare teachers to teach a complete curriculum [36]. Effective teacher professional development requires at least 50 hours of effort by a teacher [11]. We need on-line classes that support that much time investment in a structure and pace that allows teachers to fit the learning into their busy lives.

We decided to focus on ebooks as a medium because they can be self-paced. They can also be designed to provide low cognitive-load activities, which are easy to schedule in a busy life (e.g., few confounding syntax errors). We can also provide content to meet the unique needs of teachers, such as pedagogical content knowledge (PCK). Another difference between our ebooks and MOOCs is that our ebooks feature active learning using an *Examples+Practice* approach, which is explained in the next section. Much of the content in MOOCs is delivered via video lectures. A meta-analysis of 225 studies found that active learning lead to better student performance than lecture in STEM fields [17]. Active learning also had a larger effect on female students in male-dominated fields and on disadvantaged students.

2.2 Ebook Features and Educational Psychology Principles

We have presented the design of our ebook in other papers [14; 16], so we present the features at a higher-level here, with a focus on how they relate to the design principles later. The general structure of our ebooks is *Examples+Practice*. We use worked examples to teach computer science and problem solving [38], and we interleave one or two practice activities per worked example as recommended by Trafton & Reiser [40] to focus attention on the worked examples.

Our goal with the Examples+Practice format is to offer efficient and effective instruction that leads to learning across many small chunks of time. We expect the Examples+Practices sets to have lower cognitive load than traditional code writing activities, and that this lower cognitive load will lead to more learning with less time and effort [37]. A frustrating syntax error that takes hours to fix (as we know happens [6; 22]) is inefficient – there is too little learning from a large investment of time. Our hypothesis is that our ebooks will lead to high completion rates with small learning sessions (20 minutes to one hour), which will increase teachers' content knowledge and confidence.

Worked Examples: Most of our examples are in the form of *active code* segments, which are editable, executable code segments. Learners[1] are encouraged to edit these code segments, e.g., to answer a question in a practice problem. Our active code examples include *audio tours* that describe the program using audio narration. Audio tours build on the dual modality principle, that moving explanation from visual text to auditory narration can reduce cognitive load and improve learning [25; 29]. Some of our examples are in the form of a *code lens*, which is an executable code segment that uses Guo's visualization tool [18].

Practice: We have four forms of practice. Two are fairly typical: *multiple-choice* and *fill-in-the-blank questions*. The multiple-choice and fill-in-the-blank questions both offer immediate feedback tailored to the response, i.e., the learner is told more than just "right" and "wrong."

A third form of practice is a *Parsons problem* where learners are given a problem to solve and given a correct program to solve it, but the lines of the program are spread across "refrigerator magnets" which have to be dragged into the right order with the right indentation [12; 16; 20; 31]. Prior research on Parsons problems has shown a notable correlation between scores on Parsons problems and scores on code writing problems [12]. Also, the lowest quartile of students did better on Parsons problems than on code writing or code tracing problem [12], which may mean that solving Parsons problems is easier than writing code. Parsons problems provide feedback by highlighting blocks that are in the wrong place or have the incorrect indentation.

The fourth form of practice is *editing active code* segments. The feedback from this type of practice is Python error messages or the output from execution of the code.

Teacher Notes: The teacher ebook also includes pedagogical content knowledge (PCK) notes to help teachers diagnose student's misunderstandings and learn how to teach the

[1] In the ebooks described in this paper, our users are teachers, but in the role of students. We use the generic "learners" when speaking of the teachers learning through our ebook.

programming concepts. For example, one teacher note is a video that explains the student misconception that assigning one variable to another creates a relationship between the two variables, such that any change to one will automatically change the other.

Videos: Unlike MOOCs there aren't very many videos in the teacher ebook. Most of the videos demonstrate student misconceptions.

3. EBOOK ITERATIONS

All design-based research studies involve iterations of development, data gathering, and analysis. In this section, we present summaries from our previous studies and the new results of a large-scale study with teachers using version 2 of the teacher ebook. We also report on the changes we made to version 3 based on the teacher feedback from version 2.

3.1 Early Studies: Teacher Observations

We added interactive content to the ebook "How to Think Like a Computer Scientist – Interactive Edition" and observed teachers working through a chapter of that ebook. We found that the teachers didn't do all of the interactive activities. For example, they rarely watched the videos or listened to the audio tours. One of the teachers mentioned that he had watched some of the videos, but found that they covered the same content as the ebook, so he didn't feel that he needed to watch them. Some of the teachers hadn't even noticed the audio tour button and the book text never mentioned them. This study was described previously [16].

3.2 Early Studies: Log File Analysis of Student Use

A log file analysis of high school and undergraduate student use of the "How to Think Like a Computer Scientist – Interactive Edition" ebook found that most students ran the code, solved Parsons problems and answered multiple-choice questions. Fewer students edited the code, watched the videos, or listened to audio tours. While the teachers that we had observed found the Parsons problems a bit too easy, some students clearly struggled to solve them. This study was also described previously [16].

3.3 Early Studies: Usability Study

Another of our early studies concerned the usability of some of the interactive features of the ebook, which we reported on previously [14]. We tested the usability of the interactive features, active code, code lens, Parsons problems, and multiple-choice questions against those of similar ebooks (Zyante and CS Circles). Most of the teachers in the study preferred the Runestone user interface for all but one of the interactive features (the code lens).

3.4 Version 1: Teacher CSP Ebook

We conducted a pilot study with ten teachers during the spring of 2015 which we reported on in [14]. They worked through the first eight chapters of the first version of the teacher ebook at their own pace. These chapters covered variables, math operators, and assignment. The ebook introduced these concepts in several contexts: numbers, manipulating strings, making a virtual robot turtle move and draw, and modifying the colors of images. This version also contained teacher notes about common misconceptions and other pedagogical content knowledge.

The study participants were asked to take post-tests after every two chapters. They were also asked to answer feedback questions after every two chapters. Most of the teachers reported that they enjoyed the interactive features of the ebook. One teacher wrote, "*I feel like this would be an effective and beneficial tool for students and teachers*." However, a few of the teachers didn't use the interactive features very much. One of the teachers mostly answered multiple-choice questions (usually incorrectly) – she may have been rushing through the ebook to make the deadline in order to receive compensation for completing the study. Teachers who used more of the interactive features and spent more time in the chapters reported higher confidence in their ability to teach the material. Of the ten participants, five (50%) "completed" the ebook (took all of the post exams) which is a higher completion rate than is typical for MOOCs. However, this was only a small pilot study with only ten teachers, and the compensation likely influenced completion behavior.

3.5 Version 2: Teacher CSP Ebook

We made modifications based on feedback from the pilot study, such as adding a video to show how to solve a Parsons problem, asking the reader to listen to an audio tour, and breaking the chapters into smaller sections. We completed writing the ebook in the early spring of 2015. This version of the ebook contained five parts: computer abilities (chapters 1-2), naming (chapters 3-6), repeating (chapters 7-11), decisions (chapters 12-15), and data (chapters 16-19).

We evaluated this new content in several ways.

- We conducted a large scale study of teachers using the ebook on-line,
- paid a pilot CSP teacher to give us detailed feedback, and
- had a pilot CSP teacher use the ebook with her high school students and both the teacher and her students gave us written feedback.
- We used the ebook in our blended learning (partly on-line and partly face-to-face) professional development. We also made the ebook freely available to other groups offering face-to-face professional development such as the *CS Matters* group in Maryland and *Project Lead the Way*. Two of the teachers who we interviewed were part of blended learning cohorts, and that context seemed to have an influence on their use of the ebook (as seen in section 3.6.4).

3.6 Large-scale Teacher study of Version 2

We recruited teachers by sending email to our list of over 500 teachers who had attending professional development at Georgia Tech in the past. Guzdial announced the availability of the teacher ebook on his blog on April 1, 2015, with a link to more information about the study.

To qualify for the study, participants had to be at least 18 years old, hold a Bachelors degree, and could not have taught Python. Over 200 teachers applied to be part of the study from April to August 2015. While the majority of the teachers were from the United States (75%), teachers also applied from the UK, Spain, Mexico, Australia, England, Scotland, Thailand, Germany, Greece, New Zealand, Canada, France, Russia, The Netherlands, Finland, China, Pakistan, Belgium, Brazil, and the Philippines.

To qualify for the study the teachers also had to score less than 70% correct on the pretest (7 or less questions correct out of 11). The pretest consisted of multiple-choice questions on variables, assignment, conditionals, functions, lists, strings, loops, and mathematical operations. These questions were from a thesis by Juha Sorva that compiled a list of common misconceptions [35]. All multiple-choice questions included the answer "I don't know". Interestingly, several teachers registered for the study multiple times and some of them failed the pretest after passing it the first time. It appears that the teachers were deliberately failing the pretest to gain access to the ebook.

Of the 229 teachers who applied for the study, 130 teachers qualified for the study by scoring less than 70% on the pretest. Of these only 45 (35%) took the first end-of-chapter test (after chapter two) and only five people took the test after chapter 17 as shown in Figure 1.

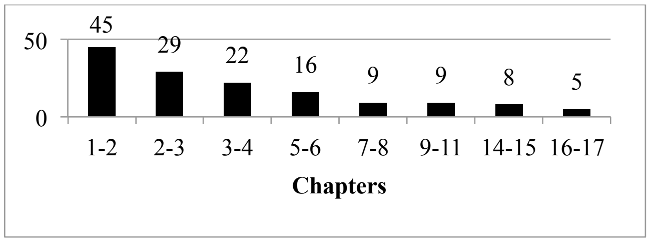

Figure 1 – The number of people who took each of the approximately every two chapter post-tests.

The completion rate for those who took the first post-test (45) to the last post-test (5) is only 11%, which is about the same completion rate as most MOOCs. However, our actual completion rate may have been higher since logging was accidentally turned off during the summer (in late June) when many teachers reported working through the ebook. Several teachers provided feedback via an external website for the later chapters, so we do think the actual completion rate was higher than 11%.

Design-based research recommends mixed methods to evaluate educational interventions and works closely with educators on the design and evaluation of educational interventions. One difficulty we faced was gathering feedback from all of the remote teachers in the study. We prompted teachers to fill out a feedback survey approximately every other chapter. The survey asked what features the teachers found most valuable, what they would change to make the readings more effective, and whether the chapter covered the content well. We initially sent email to teachers who were part of the ebook study to request that they provide feedback when the log file showed that they had completed about every other chapter. In late June we added links to the external feedback surveys directly in the ebook at the end of about every other chapter. Even though we are missing the log file data from late June to fall, we do have teacher feedback on what the teachers found valuable and what should be improved.

3.6.1 *Teacher Feedback*

Thirty-eight teachers filled out a total of 74 feedback forms during the study. A count of the number of times each feature was mentioned as valuable is shown in Figure 2 below.

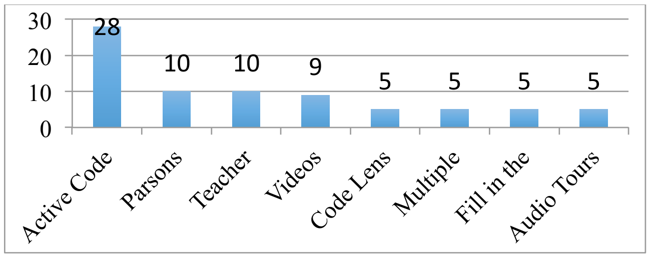

Figure 2– The number of times each feature was mentioned as valuable in the feedback

We computed the number of times each feature was mentioned. Some of the comments mentioned more than one feature. Some comments were vague such as simply mentioning "practice" which we interpreted as all the types of practice problems: multiple-choice, Parsons problems, and fill in the blank.

3.6.1.1 *Feedback on the Content*

Some teachers commented on the content of the ebook. *"I liked the pace and the quick introduction of turtle and images.* Several teachers give general comments: *"So far I love the ebook. It is well organized, [has] appropriate links, [and is] not too overwhelming for a new coder."* Another teacher wrote, *"I think the explanations are very clear and easy to understand."*

3.6.1.2 *Feedback on the Worked Examples*

The active code feature was the feature that was most often mentioned as valuable (28 times) as shown in Figure 2. One teacher commented, *"The interactive code sections (active code) were very helpful. I think my students would benefit from using this format because they are able to see immediately whether their code works and get feedback."* Teachers also appreciated being able to edit the code in the active code feature. *"I really enjoy being able to manipulate the code. I do have some programming experience and it's fun to play with the numbers and watch the outputs change."* Audio tours were mentioned five times. One teacher wrote, *"The audio tours are a great idea. Students hear the vocab being used correctly and can repeat it if necessary."* The code lens, which gave them the ability to step through code and see the values of the variables change, was also mentioned 5 time with one teacher writing, *"Using the code lens tool for tracing was absolutely fantastic."*

3.6.1.3 *Feedback on the Practice Problems*

One teacher wrote that the most valuable feature was *"The questions that examine your understanding."* Parsons problems were mentioned 10 times, while multiple choice and fill in the blank questions were each mentioned 5 times. Teachers also liked that some of the practice problems required them to modify the example code such as changing the "if" statements to "if "and "else".

3.6.1.4 *Feedback on Teacher-Focused Features*

The teacher notes were mentioned 10 times. *"It was helpful that you showed the common misconceptions students have with variables. That should help me address those directly as I teach."* Another teacher wrote *"I appreciated the note about the rainfall problem -- anticipating students' struggles and giving us real data about how student perform on the task."* Several teachers mentioned the short videos in the ebook as being valuable. *"Videos are great."* Some teachers appreciated the end of chapter summaries. One teacher wrote, *"I like the key terms provided at the end in the form of a summary review."*

Not all teachers were happy with the ebook. One teacher commented, *"None of it [was valuable]. I already know this stuff."* While we intended this ebook for teachers with no textual programming experience, several teachers had much more experience than we expected of participants in the study.

3.6.1.5 *Teacher Suggestions for Changes*

Several teachers wanted the ability to write more code in the ebook and thought that some of the examples would be too difficult for beginners. *"Perhaps have the reader write some very simple code along with the examples you already have. If this is for true beginners, some of your examples are going to be total 'Greek' to them, even with the explanations. The current examples may be a bit intimidating."* Some people found errors in the ebook, *"there were a few typos"* or in the user interface, *"Sometimes the audio did not work."* Some teachers found the order of the first two chapters strange. *"The sequence seemed a little goofy to start with coding [in chapter 1] and then go into what a computer can do in chapter 2]."*

Some teachers found particular chapters and/or concepts difficult. One teacher wrote after chapters 7 and 8, *"I think at this point I would like to see MORE examples. This was the first chapter that I felt in over my head. I would have liked to also see examples of a FOR loop and a WHILE loop that does the same thing side by side – so that I could compare the two and see what the differences are. I would like to see a list of CODES LEARNED and what they do, and the proper syntax, after each chapter. That way I would be constantly reviewing both the CODES (Python commands) and the CONCEPTS (vocabulary)."* One teacher suggested that we, *"elaborate [more on the] explanation of functions."* Another teacher wrote *"Chapter 16 [the chapter introducing data analysis] was very difficult."*

Teachers also wanted more *"more quiz questions"* and *"more examples"*, and they wanted more answers to the practice questions to check themselves. *"It would be great if we were given a way to access the answers to different exercises we've done. That way, when we've finished, we can check our work. Or, if we have no idea how to start, we can get an idea of how to begin."* Teachers also asked for hints to help them when they were stuck, especially for the Parsons problems. *"Maybe give more hints when you get things wrong in a drag and drop or a program construction task. I got stuck, didn't know what was wrong - and there was nothing to direct me."*

Teachers also wanted the ebook to cover more of the content of the CS Principles course. It currently covers only two of the big ideas from the CS Principles course: programming and data.

3.6.1.6 Teacher Confidence

Since one of our goals was to increase teacher confidence in his or her ability to teach the content, we were particularly pleased to find evidence of increased confidence. *"I have been told that I will teach Computer Science next year – and I am completely overwhelmed and intimidated – but this course is helping to put my mind at ease."* In a later chapter the same teacher said, *"I feel as if I am slowing adding on to my knowledge of the Python language. It is helpful to 'build' my knowledge."*

3.6.2 Feedback from the Non-Completers

In the fall of 2015 we asked those who hadn't completed the ebook why they did not complete it. Seven of the nine (78%) teachers said that they didn't have time due to other commitments or family issues, as expected from Benda's study [6].

Two of the nine (22%) gave answers based on the content of the ebook. One teacher wrote, *"I found the later lessons/exercises were less relevant to the sort of teaching I deliver in programming."* Another teacher listed the reasons as *"the user interface of the book; redundancy of activities; too much jumbled info on the screen; did not pick up where I left off."* When asked what most needed to be changed two teachers mentioned some way to keep track of where you were, *"It wasn't at first apparent which lessons I had completed, although I know that later this functionality was added."* This functionality was added during the summer of 2015 in response to this feedback.

Other teachers were concerned about the interface. *"I would like to see it look more like an e-zine. Would give it a more professional feel."* Another teacher was frustrated by the lack of help, *"Sometimes if I was stuck I didn't know how to complete a task and there was no help."* Some of the feedback was contradictory with one teacher asking for more videos and another suggested that we *"lose the videos."*

Teachers reported working on the ebook afterschool, in the evenings, and in the summer. The amount of time they reported spending in a session ranged from 20 minutes to 1.5 hours. When asked what would have encouraged them to finish the ebook most of them said more time. They also suggested face-to-face professional development, continuing-education credit, financial incentives, certificates, and feedback on their performance on the assessments.

3.6.3 Feedback from the Completers

In the fall of 2015 we also asked those who had completed the ebook to fill out a survey. We asked how much total time they spent on working through the ebook, and the answers ranged from six to 30 hours with most answering around 10-20 hours. As we mentioned earlier logging was accidently turned off in late June so we can't determine the total time spent from the log files. The reasons for completing the ebook included it being required, wanting to learn the material, trying to prepare for teaching CSP in the fall, and *"I like to finish things."*

When asked what should be added one teacher wanted *"supporting lesson plans."* This teacher was clearly interested in the ebook as a resource for *students*, not for teacher development. *"The ebook is designed so that students can independently work on activities. There needs to be a way for a teacher to hold students accountable for completing the ebook. Are there assignments? Are there quizzes?"* Another teacher wrote, *"Enjoyable. Work[ed] at my own pace. Thought provoking and engaging. I understand CSP and Python better than I did when I began. I could use this to teach in my class if it was possible."* Another teacher wanted, *"more questions in each section and more opportunities to write code."* Asked about the overall experience one teacher reported, *"I thought it was a good experience overall. But I did get very frustrated at certain points because I was not understanding it."*

3.6.4 Interviews with Teachers

To gather more in-depth feedback from the study participants our evaluators interviewed three of the teachers from the United States during the winter of 2016. Each interview was conducted over the phone, lasted between 50-60 minutes, and was audio recorded and transcribed.

Teachers' responses were first sorted into broad coding categories. The coding categories and themes were guided by a set of interview questions and also emerged iteratively from the data. Data analysis proceeded by moving back and forth between individual cases and the more general view across cases. The individual cases were then used as examples of the more general coding categories (similar to the approach used in [30]).

Table 1. Teacher id, experience, motivation, and number of completed chapters

ID	Experience	Motivation	Completed
A	First time teaching CS	May teach CSP in the fall	13 chapters
B	Teaching AP CS A for 10 years	Students want to learn Python for robotics competition	3 chapters
C	Taught an introductory computing course using Scratch	May teach CSP in the fall	8 chapters

Teacher A, had been told to complete a certain number of chapters before her in-person professional development started "*I took a Computer Science training class this summer. Before the class started, they asked that we go through this book. They asked us to get to a certain point in the book. I got actually a little bit past where they had asked us to go*."

Teacher B was evaluating the ebook as a possible resource. *"I tried to sign up for every single course that I possibly could to get me familiarized with the material. So, basically, I've used [the] eBook for more of a resource for me. I'm really still in the process of trying to kind of just figure out what I need for this unit, or whatever it is I need to do for my courses."*

Teacher C reported that she had signed up for several professional development opportunities at the same time and stopped working with the ebook when the workload became too much, *"I thought I could do both… the CSP Mobile [and] CSP Python. So, I was also in the CSP Mobile Course while I was doing the Python. Then I stopped taking the Python [eBook] course."*

Interestingly, all three teachers used the eBook at school. They worked between 30 and 60 minutes at a time. When asked how they used the ebook, all three of the interviewed teachers mentioned reading the text, completing the practice items (multiple choice, fill-in-the-blank, and Parsons problems), and completing the end of chapter tests. One teacher said that even though she was asked to edit the code and use the code lens to step through code, she didn't do that. *"I would do all the questions just to make sure I understood it. I didn't really play around with the code that much, which in retrospect, I think … would have been better if I did. Like… you could run the code. Then they said you could change things and play with the code. But I really didn't. That was more of a time thing. … I did not use the code lens that often… I understood what I was doing and I didn't need it"*.

Teacher B mentioned using the metaphors from the eBook in her computing class and that the eBook helped her generalize computing concepts. *"I found the different metaphors for a variable that I can use for teaching for Python or for any other language. The computer's contents are taught in such a way that it's not just I'm learning Python. I'm also learning computer concepts and how to teach them and apply them to any language."* She also said, *"I think everything was useful. I really do. I really like this eBook."*

All three teachers reported that the ebook contributed to their knowledge of and confidence in teaching CS. Teacher C was asked how much the ebook contributed to her confidence and she replied, *"Quite a bit really. I mean, I probably would have been lost without it. When I took the professional development workshop and was introduced to the ebook at the beginning, I really was lost."* She mentions feeling that professional development instructors were initially *"talking above my head."* However, she says that the ebook offered her scaffolds to better understand the face-to-face professional learning: *"I mean, I know that sounds awful, but it was just like overwhelming. Then as I went on throughout the summer and I took some other courses and as I was exposed to ebook, I could understand things."*

Teacher A wanted more opportunities to write code from scratch. *"I still think it would have been more helpful if in sections they would have prompted me to write my own code and me have to figure it out, or if I get stuck, be able to contact someone and say, 'I can't get this code to work.' That kind of thing instead of just always answering the multiple-choice questions or doing the drop and drag (Parsons problems)."*

All three teachers mentioned that they plan to return to the eBook in the future.

3.6.5 Feedback from CSP Pilot Teachers

For more detailed feedback, we paid a CSP pilot teacher in Georgia to review the teacher ebook, which she did in the spring of 2015. She found errors in the ebook, told us where she thought we needed more exercises or better descriptions, and also suggested adding answers for some of the more difficult practice problems to reduce frustration for teachers who were new to programming. These changes were done to version 2 in June before our in-person teacher professional development at Georgia Tech.

Another pilot CSP teacher in Georgia used version 2 of the teacher ebook with her students and gave us feedback on which chapters needed additional information. She said, *"I loved the data unit at the end."* She made specific recommendations on additional material that should be added to particular chapters. For example, she asked that the chapter on strings include a note to highlight the fact that printing a variable prints the value of the variable and printing a string prints the exact characters in the string. She also recommended more content for some of the introductory chapters based on how she was using the ebook with her students in class.

3.6.6 Log File Analysis

We conducted an analysis of the log file from before logging was accidently turned off in late June of 2015 to gather more evidence about how the teachers actually used the interactive features.

Figure 3 below shows how many of the teachers, who completed a particular section of chapter three (on the use of variables with numeric values), did each of the interactive activities in the section. The largest number of teachers (66) used the first code lens (black) and a large number also ran the first code (blue) using the active code feature. However, use of these features mostly decreased from the beginning to the end of this section. A large percentage (77%) of the teachers watched the videos (purple), attempted Parsons problems (73%) (green), and answered multiple-choice questions (74%) (gold). The number of teachers who used these last three features remained fairly steady over the course of this section. However, only 7 (11%) to 11 (16%) of the teachers listened to the audio tours (red), even though this chapter explicitly directed the reader to listen to one of the audio tours.

Table 2: Color to Activity Legend for Figure 3

Black	Code lens
Blue	Active Code Run
Gray	Active Code Edit
Red	Audio Tour
Gold	Multiple Choice Question
Green	Parsons Problem
Purple	Video

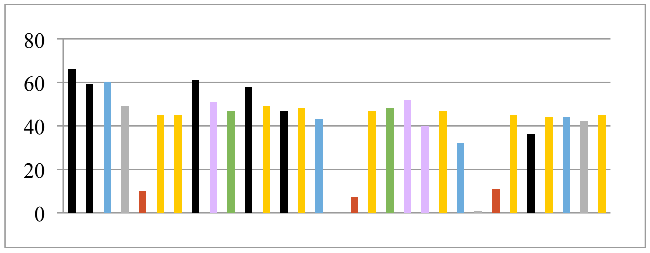

Figure 3. Num. who did Each Activity in Part of Chapter 3

The number of unique teachers who did each of the activities in chapter 10 (Repeating Steps with Turtles) appears in Figure 4. The analysis shows that most of teachers ran the active code (blue), solved Parsons problems (green), and answered multiple-choice questions (gold). As seen before, very few teachers edited the code, except for the fifth active code in the chapter, which would not run until the user edited the code. In this chapter only 1-2 (6% to 11%) of the teachers listened to the audio tours.

One of our questions was, "How much do teachers actually engage with code in the ebook?" Consistent with teacher self-report (Figure 2), log file data shows that most teachers ran the code. Surprisingly, few teachers actually *edited* the code, even when they were directed to by the ebook. One of the teachers that we interviewed said that she answered the questions and if she got those correct, she didn't feel the need to actually edit the code. Some teachers reported the code lens visualizer as being very valuable in their feedback. However, one of the interviewed teachers said she didn't bother using the code lens since she already understood the code.

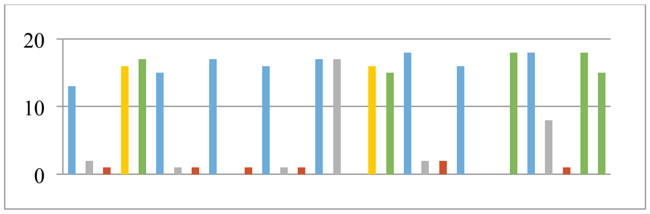

Figure 4. Unique Users who did Each Activity in Chapter 10

Some teachers listened to the audio tours as a supplement to reading the program code and did report finding them valuable in their feedback surveys. However, the log files show that the audio tours are not used as much as running the code and doing the practice activities. However, audio tours may particularly help new computer science teachers by modeling how to talk about code.

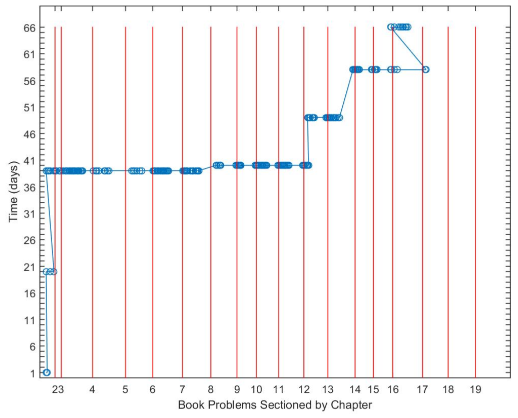

Figure 5. Teacher A's progression through the ebook

We reviewed how one of the teachers who we interviewed progressed through the ebook for additional insight into her behavior. Figure 5 shows numbered days of the study on the vertical axis, and the sequence of activities in the book (by chapter) horizontally, with a mark for each activity attempted. The teacher started with a few interactions in chapter 1, then about 20 days later did a few more actions in chapter 1 and then about another 20 days later she began to work through the ebook more consistently. She then had some long sessions working through many chapters in the same session. As she said in her interview, she was told to work through a certain number of chapters before her in-person professional development, which may be why she suddenly seemed to focus on completing many chapters quickly.

3.7 Version 3: TeacherCSP Ebook

We ended the teacher study in September of 2016. Based on the feedback from the teachers we added the following.

- End of chapter summaries of the Python functions
- Additional material to the chapters of the ebook that teachers had pointed out as needing more examples and better explanations. For example we added a side-by-side comparison of a **while** and **for** loop, and additional examples using **while** loops.
- We added 10 end-of chapter exercises, even though that broke our *Examples+Practice* format. We added at least one exercise per chapter that required the teacher/student to write all of the code from scratch. Each of the exercises included the answer in a separate tab. An additional tab was added with a link to a discussion forum so that teachers could discuss the question.

Also in response to teacher requests, we released a companion student ebook in the fall of 2015 that has the same content, but removes the answers and pedagogical content knowledge notes. We are pleased to note that several of the teachers from the teacher study used the ebook with their high school students during the 2015-2016 academic year. They have also told us that they intend to use it again next year.

We haven't analyzed the data from version 3 of the teacher ebook yet. However, we are pleased that some of the teachers from the version 2 teacher study have continued working with the version 3 teacher ebook.

4. DESIGN PRINCIPLES

The design principles listed here are the ones that we have the most confidence in based on our multiple iterations, teacher observations, teacher feedback, teacher interviews, log file data analyses, and laboratory experiments. Evidence from the ebook iterations and experiments supports our belief that building upon educational psychology design principles is the right first step in developing our ebooks. However, we still don't completely understand what makes learning in computer science challenging. We cannot assume that the educational psychology principles will work as predicted. We have to test, and sometimes modify our approach, because of the unique challenges of learning computing.

4.1 Use Worked Examples + Practice

The teacher feedback provides evidence that the teachers appreciated the interactive nature of the ebook with worked examples in the form of active code or code lens paired with multiple-choice, fill in the blank or Parsons problems. The log file analysis shows that the majority of the teachers ran the examples and did the practice problems.

4.1.1 Use Subgoal Labels

As we started this project, a collaboration with Psychology professor Richard Catrambone and Ph.D. student Lauren Margulieux led to studies supporting the belief that subgoal labeling of worked examples facilitated student learning, retention, and transfer [24]. A follow-up experiment showed that the effect was twice as strong for teachers as for participants drawn from the undergraduate psychology pool [23]. Morrison continued the experiments and showed that the subgoal labeling effects extended to C-like textual languages [27].

It can be challenging to invent good subgoal labels, as has been noted in the literature on worked examples [2]. We do not use subgoal labeling on all examples. We do not use the *same* subgoal labels on all program examples. When we do use them, we use them consistently across a chapter.

Morrison and Margulieux investigated different ways of using subgoal labels (e.g., giving students the labels compared with asking students to fill in the blank to construct subgoal labels) on program examples. Their results were contrary to the results predicted by previous literature [40]. They hypothesize that the implicit cognitive load of program understanding was so high that it swamped different uses of subgoal labeling (which might also explain the modality results). When they used a more sensitive learning measure in their subgoal labeling experiment (based on Parsons problems), they got results that matched the educational psychology predictions [34].

4.1.2 Use Low-Cognitive Load Practice Problems

Evidence from the teachers' feedback and the log file analyses show that teachers are using the low-cognitive load practice problems. Parsons problems, in particular, were mentioned as being valuable 10 times in the feedback, compared to five times for multiple choice or fill in the blank questions. Log file analysis also shows that Parsons problems were used as much as the multiple-choice questions and running code and far more than editing code. The most recent set of experiments also support the belief that Parsons problems have lower cognitive load than the same practice problem as a code-writing activity, and that subgoal labels also improve performance on Parsons problems [26].

4.1.3 Maybe Provide Audio Tours

Morrison attempted to measure the benefits of audio tours in an experiment where she compared textual, auditory, and combined text plus narration explanations of program code. However, she found no difference between the three conditions. Another experiment in the research literature also failed to find a difference between modality conditions on explanations of program code [34]. However, feedback from the teachers provides evidence that at least some teachers found the audio tours valuable. Yet, the log file analysis showed that only a small percentage of teachers actually listened to the audio tours. Audio tours are probably most beneficial for the teachers who have not taught programming before.

4.2 Provide lots of content

The teacher feedback included many requests for more content. They want more examples, more exercises, and more coverage of the CS principles course. They want answers to all the practice problems and exercises. In addition, teachers also asked for external resources such as lesson plans, quizzes, pacing guides, and project ideas.

4.3 Provide what teachers expect

Part of the answer to what teachers need is not about cognition, but about teacher expectations.

- Teachers wanted end-of-chapter exercises, even if our theoretical framework recommends pairing Examples+Practice. We recommend providing end of chapter summaries of both the computing concepts and Python procedures and functions covered in the chapter. Teachers explicitly asked for this and several teachers mentioned the concept summaries as being useful.
- Part of what we are testing with our ebook is how much programming teachers can learn and how much we can improve their confidence in their ability to teach programming *without* requiring them to do significant amounts of programming. However, teachers studying computer science expect to program, and some expect to code from scratch.

We ignore these requests at the risk of losing participation.

4.4 Support teachers understanding code

Reading and understanding programs is a challenging task, especially for novices. It's also an important task since it's one of the most common activities of expert CS teachers [30]. The teacher feedback shows that teachers found the code lens and audio tours useful for improving their understanding of code. Evidence from controlled experiments also shows that subgoal labels improve learners understanding of code.

4.5 Save teachers time

Teachers don't have a great deal of time for professional development. They often work in relatively small chunks of time. Add features to save teachers time.

- Provide a way to mark a section as completed and a way to return a teacher to where he or she was last.
- Break the material into chapters and sections. Make each section short enough to allow a teacher to complete it in 15-20 minutes, which makes it easier to fit into the day and to schedule.
- Provide answers for all the practice problems to help reduce teacher frustration and to scaffold teachers with no prior programming experience.

5. LIMITATIONS

Since logging was accidently turned off from late June till fall 2015, we don't have all the data from the teachers' use of version 2 of the ebook. In particular we don't have the end of chapter tests for many of the teachers who worked through the ebook, so we can't comment on how much the teachers learned from using the ebook. In addition, we are basing some of our recommendations on teacher feedback, however we only received feedback from thirty-eight teachers.

6. CONCLUSIONS

The teacher feedback and interviews serve as an existence proof that an Examples+Practice ebook approach (versus a MOOC-based, or in-class coding-centric approach) can achieve our goal of greater teacher self-confidence. We do not yet know how to design so that *all or most* teachers come away with increased self-confidence.

The contribution of this paper is a set of design principles that others can use when developing interactive ebooks for computer science teachers. We have presented evidence to support using a worked examples plus practice approach. The ebook provided worked examples using the active code and code lens features, and practice problems using multiple-choice, Parsons problems, and fill-in-the-blank questions.

7. ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. 1432300. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] Astrachan, O., Briggs, A., Diaz, L., and Osborne, R.B., 2013. CS principles: development and evolution of a course and a community. In *Proceedings of the Proceeding of the 44th ACM technical symposium on Computer science education* (Denver, Colorado, USA2013), ACM, 2445382, 635-636.

[2] Atkinson, R.K., Derry, S.J., Renkl, A., and Wortham, D., 2000. Learning from Examples: Instructional Principles from the Worked Examples Research. *Review of Educational Research 70*, 2, 181–214.

[3] Barab, S. and Squire, K., 2004. Design-Based Research: Putting a Stake in the Ground. *The JOURNAL OF THE LEARNING SCIENCES 13*, 1, 1-14.

[4] Bell, T., Andreae, P., and Lambert, L., 2010. Computer Science in New Zealand high schools. In *Proceedings of the Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103* (Brisbane, Australia2010), Australian Computer Society, Inc., 1862223, 15-22.

[5] Bell, T., Andreae, P., and Robins, A., 2014. A Case Study of the Introduction of Computer Science in NZ Schools. *Trans. Comput. Educ. 14*, 2, 1-31.

[6] Benda, K., Bruckman, A., and Guzdial, M., 2012. When Life and Learning Do Not Fit: Challenges of Workload and Communication in Introductory Computer Science Online. *Trans. Comput. Educ. 12*, 4, 1-38.

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

[8] Brown, N.C.C., Kolling, M., Crick, T., Jones, S.P., Humphreys, S., and Sentance, S., 2013. Bringing computer science back into schools: lessons from the UK. In *Proceedings of the Proceeding of the 44th ACM technical symposium on Computer science education* (Denver, Colorado, USA2013), ACM, 2445277, 269-274.

[9] Brown, N.C.C., Sentance, S., Crick, T., and Humphreys, S., 2014. Restart: The Resurgence of Computer Science in UK Schools. *Trans. Comput. Educ. 14*, 2, 1-22.

[10] Caspersen, M.E. and Nowack, P., 2013. Computational thinking and practice: a generic approach to computing in Danish high schools. In *Proceedings of the Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136* (Adelaide, Australia2013), Australian Computer Society, Inc., 2667214, 137-143.

[11] Darling-Hammond, L., Wei, R.C., Andree, A., Richardson, N., and Orphanos, S., 2009. Professional learning in the learning profession. *Washington, DC: National Staff Development Council.*

[12] Denny, P., Luxton-Reilly, A., and Simon, B., 2008. Evaluating a New Exam Question: Parsons Problems. In *Proceedings of the International Computing Education Research Conference* (Sydney, Australia2008), ACM.

[13] Ericson, B. and Guzdial, M., 2014. Measuring demographics and performance in computer science education at a nationwide scale using AP CS data. In *Proceedings of the Proceedings of the 45th ACM technical symposium on Computer science education* (Atlanta, Georgia, USA2014), ACM, 2538918, 217-222.

[14] Ericson, B., Moore, S., Morrison, B., and Guzdial, M., 2015. Usability and Usage of Interactive Features in an Online Ebook for CS Teachers. In *Proceedings of the Proceedings of the Workshop in Primary and Secondary Computing Education* (London, United Kingdom2015), ACM, 2818335, 111-120. DOI= http://dx.doi.org/10.1145/2818314.2818335.

[15] Ericson, B.J., Guzdial, M., and Mcklin, T., 2014. Preparing secondary computer science teachers through an iterative development process. In *Proceedings of the Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (Berlin, Germany2014), ACM, 2670781, 116-119. DOI= http://dx.doi.org/10.1145/2670757.2670781.

[16] Ericson, B.J., Guzdial, M.J., and Morrison, B.B., 2015. Analysis of Interactive Features Designed to Enhance Learning in an Ebook. In *Proceedings of the ICER* (Omaha, NE, USA, August 09-3, 2015 2015), ACM. DOI= http://dx.doi.org/http://dx.doi.org/10.1145/2787622.2787731.

[17] Freeman, S., Eddy, S.L., Mcdonough, M., Smith, M.K., Okoroafor, N., Jordt, H., and Wenderoth, M.P., 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academies of Science 111*, 23, 8410-8415

[18] Guo, P.J., 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education* ACM, 579-584.

[19] Guzdial, M., 2015. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics 8*, 6, 1-165.

[20] Helminen, J., Ihantola, P., Karavirta, V., and Malmi, L., 2012. How Do Students Solve Parsons Programming Problems? - An Analysis of Ineraction Traces. In *Proceedings of the International Computing Education Research Conference* (Aukland, New Zealand2012), ACM, 119-126.

[21] Hubwieser, P., Magenheim, J., M, A., #252, Hling, and Ruf, A., 2013. Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the Proceedings of the ninth annual international ACM conference on International computing education research* (San Diego, San California, USA2013), ACM, 2493395, 1-8.

[22] Jadud, M.C., 2006. *An exploration of novice compilation behaviour in BlueJ*. University of Kent.

[23] Margulieux, L.E., Catrambone, R., and Guzdial, M., 2013. Subgoal labeled worked examples improve K-12 teacher performance in computer programming training. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, 978-983.

[24] Margulieux, L.E., Guzdial, M., and Catrambone, R., 2012. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Proceedings of the ninth annual international conference on International computing education research* ACM, 71-78.

[25] Mayer, R.E. and Moreno, R., 1998. A split-attention effect in multimedia learning: Evidence for dual processing systems in working memory. *Journal of Educational Psychology 90*, 2, 312.

[26] Morrison, B.B., Margulieux, L.E., Ericson, B., and Guzdial, M., 2016. Subgoals Help Students Solve Parsons Problems. In *Proceedings of the Proceedings of the 43rd ACM technical symposium on Computer Science Education* (Memphis, Tennessee2016).

[27] Morrison, B.B., Margulieux, L.E., and Guzdial, M., 2015. Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* ACM, 21-29.

[28] Morrison, B.B., Ni, L., and Guzdial, M., 2012. Adapting the disciplinary commons model for high school teachers: improving recruitment, creating community. In *Proceedings of the ninth annual international conference on International computing education research* (Auckland, New Zealand2012), ACM, 2361287, 47-54. DOI= http://dx.doi.org/10.1145/2361276.2361287.

[29] Mousavi, S.Y., Low, R., and Sweller, J., 1995. Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology 87*, 2, 319.

[30] Ni, L., 2011. Building professional identity as computer science teachers: supporting secondary computer science teachers through reflection and community building. In *Proceedings of the seventh international workshop on Computing education research %@ 978-1-4503-0829-8* ACM, Providence, Rhode Island, USA, 143-144. DOI= http://dx.doi.org/10.1145/2016911.2016942.

[31] Parsons, D. and Haden, P., 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education* (Hobart, Australia2006), Australian Computer Society, Inc., 1151890, 157-163.

[32] Sadler, P.M., Sonnert, G., Coyle, H.P., Cook-Smith, N., and Miller, J.L., 2013. The influence of teachers' knowledge on student learning in middle school physical science classrooms. *American Educational Research Journal 50*, 5, 1020-1049.

[33] Smith, M., Computer Science for All. , https://www.whitehouse.gov/blog/2016/01/30/computer-science-all, Accessed 2016, July 13

[34] Solomon, H.M., 2005. *The effect of audio narration in computer-mediated instruction on procedural fluency by students of varying reading levels*. Florida State University.

[35] Sorva, J., 2012. *Visual program simulation in introductory programming education*. Aalto University.

[36] Spradling, C., Linville, D., Rogers, M.P., and Clark, J., 2015. Are MOOCs an appropriate pedagogy for training K-12 teachers computer science concepts? *J. Comput. Sci. Coll. %@ 1937-4771 30*, 5, 115-125.

[37] Sweller, J., 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science 12*, 2, 257-285.

[38] Sweller, J. and Cooper, G., 1985. The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. *Cognition and Instruction 2*, 1, 59-89.

[39] Taylor, K. and Miller, C.C., De Blasio to Announce 10-Year Deadline to Offer Computer Science to All Students, http://www.nytimes.com/2015/09/16/nyregion/de-blasio-to-announce-10-year-deadline-to-offer-computer-science-to-all-students.html, Accessed 2016, July 13

[40] Trafton, J.G. and Reiser, B.J., 1993. The contributions of studying examples and solving problems to skill acquisition. In *Proceedings of the Proceedings of the 15th Annual Conference of the Cognitive Science Society* (Hillsdale, NJ1993), Lawrence Erlbaum Associates, Inc., 1017–1022.