2015

Usability and Usage of Interactive Features in an Online Ebook for CS Teachers

Barbara Ericson
Georgia Institute of Technology

Steven Moore
Georgia Institute of Technology

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

Mark Guzdial
Georgia Institute of Technology

Usability and Usage of Interactive Features in an Online Ebook for CS Teachers

Barbara Ericson and Steven Moore
College of Computing, Georgia Tech
801 Atlantic Ave, Atlanta, GA 30332–0280
ericson@cc.gatech.edu,
StevenJamesMoore@gmail.com

Briana Morrison and Mark Guzdial
School of Interactive Computing, Georgia Tech
85 5th Street, NW, Atlanta, GA 30332–0760
bmorrison@gatech.edu,
guzdial@cc.gatech.edu

ABSTRACT

There are too few secondary school computing teachers to meet international needs for growing secondary school computing education. Our group has created an ebook to help prepare secondary teachers to teach the programming and big data concepts in the new AP Computer Science Principles course. The ebook was designed using principles from educational psychology, specifically worked examples and cognitive load. The ebook interleaves worked examples and interactive practice activities, which we believe will lead to more efficient and effective learning than more typical approaches to learning programming. This paper reports the results from initial studies of our ebook. First, we conducted a usability study comparing three different ebook platforms. Next, we conducted a study of teacher use of the ebook. Ten teachers worked through the first eight chapters of the ebook at their own pace. Five of the ten teachers completed the first eight chapters which is a 50% completion rate. Significantly, teachers who used more of the interactive features in the ebook did better on the post-tests and reported higher confidence in their ability to teach the material than teachers who used few of the interactive features.

CCS Concepts

•Applied computing → Interactive learning environments;

Keywords

Online education; high school teachers; secondary education; professional development; computer science principles;

1. THE NEED FOR ONLINE TEACHER EDUCATION

There are too few secondary school computing teachers to meet international needs for growing secondary school computing education [5, 6]. We have had good success with using face-to-face professional development to increase the number and quality of secondary computing teachers, but it's expensive to do well. Our Disciplinary Commons for Computing Education had positive results, and met for several hours on a Saturday once a month for most of an academic year [25]. The US CS10K effort aims to prepare 10,000 teachers to teach introductory computing courses at the secondary level [1, 2]. We would be challenged to even provide physical space and time for teaching 10,000 new CS teachers. Economically, an online component would be helpful – perhaps even necessary.

Our effort is aimed towards providing high-quality computing education that meets the needs and prior knowledge of potential teachers, in an online setting that can fit into their lives. From our previous work studying professionals taking online computer science (CS) classes [3], we know that success or failure in the learning opportunity is often determined by whether the learning activities can fit into their busy lives. The most common failure mode for adult professionals taking online CS courses was that something in their home lives took more of their time, and they were never able to catch up in the course.

The goal of making the learning activity fit into their lives has two significant goals:

- The activity has to be *usable* so that time is spent on fruitful activities, not on figuring out the user interface.
- The activity has to be *efficient* so that the learning benefit is worthwhile for the time spent. The efficiency goal has three operational parts:
  1. Teachers have to learn something worthwhile.
  2. The activity has to lead to learning.
  3. The benefit must be high enough and cost low enough that teachers actually complete the activity.

Given the importance of preparing more high school teachers internationally, and the economic necessity to provide online options, we see that it is important to find alternative options to MOOCs (Massive Open Online Courses). Completion rates are low for MOOCs [30, 20] and have been reported around 5–10% for CS teacher MOOCs [14, 32]. While "completion" may not be necessary for students who are simply "browsing" (one of the categories of MOOC student intentionality in Reich's study [30]), completion matters when teachers are learning a curriculum that they have

WiPSCE '15, November 09-11, 2015, London, United Kingdom

DOI: http://dx.doi.org/10.1145/2818314.2818335

never taught before. We need teachers to learn all the concepts that they are responsible for teaching.

Our group has created an ebook to help prepare secondary teachers to teach the programming and big data concepts in the new Computer Science Principles course that is part of the CS10K effort [1, 2]. We focus on these concepts as they are among the most difficult to learn and the most critical for developing CS teacher confidence and identity [27, 28]. We draw upon the developing research on how to build ebooks for teaching computer science [21]. Our ebook was designed using principles from educational psychology, with particular attention to worked examples and cognitive load. The ebook interleaves worked examples and practice activities, based on research on the optimal pattern [34]. We believe that our ebook will lead to more efficient and effective learning than the more typical apprenticeship approach to learning programming. We believe that our ebook will lead to higher completion rates than those reported for teachers in CS MOOC's because of the book's efficiency.

In this paper, we report on the results from two initial studies of our ebook. First, we conducted a usability study where we asked 18 teachers to compare three different ebook platforms (to one another, and not to a static textbook as in Edgcomb et al [9]). We found that our platform was highly usable. Next, we conducted a study of teacher use of the ebook as they worked through the first eight chapters of the ebook at their own pace. In our study, we were looking at the features the teachers used, how they used the features, and if there was any evidence of learning based on the feature use.

The point of our ebook effort is to identify design guidelines for successful computer science teacher professional learning media. We use our ebook as a platform to test design guidelines. This paper is an exploration of the effectiveness of our design decisions.

2. DESIGN OF THE EBOOK

We built our ebook using the Runestone Interactive platform [24]. Runestone ebooks are accessed through a Web browser. They have sections and chapters, like a traditional book. Each chapter appeared as a single web page to the participants. While the ebooks can have videos embedded in them, they are not structured around video lectures, in contrast to a lecture-based MOOC [32, 8, 12].

In our study of professionals taking online CS courses, we found that the apprenticeship model for CS learning made it difficult for professionals to adequately predict how much time the course would take [3]. As we know from previous work in computing education research (e.g., [19]), students often spend hours struggling with syntax and inscrutable error messages. As one participant told us, "There were times that it would take me hours to find one comma out of place, or find that one something that was wrong, so I didn't mind sticking with it but it just got to the point where I just didn't get it." We chose an ebook model as an online version of an understandable and predictable medium, a book. Teachers already know how to fit reading a book into their lives, e.g., a page here, a couple pages there. We built interactive elements to fit into those small chunks of time with a predictable time cost.

Runestone books contain interactive elements, which allow them to go beyond simple PDF or EPUB ebooks. Users can edit and execute Python programs within the pages of the ebooks. They can step forward and backward through visualizations of code, using Philip Guo's visualizer [11]. The platform offers multiple-choice and fill-in-the-blank questions as practice opportunities. The platform includes support for highlighting text (and saving those highlights), and it remembers where you were last in the ebook to return you to that page when you come back.

We extended the platform in three significant ways:

- We provided support for *Parsons Problems* [29] where students re-arrange blocks of code to construct solutions to challenges. We used the implementation from Juha Helminen [17].
- We added *Audio Tours* to the programming problems. From eye-tracking studies, cognitive science research has shown that programs are read more like diagrams than like prose [16]. The modality principle in educational psychology design explains why audio narration facilitates comprehension of a diagram better than a text description accompanying the same diagram [26, 23]. For each of the programs in our ebook, we include an audio description of the program (Figure 1).
- We provide support for *groups of readers ("book clubs")* to read the book together. A group of readers can discuss the book, negotiate a schedule for completing chapters, and see each other's progress through the book (Figure 2). However, our support for reading groups was not working correctly during this study.

Figure 1: The Audio Tour interface

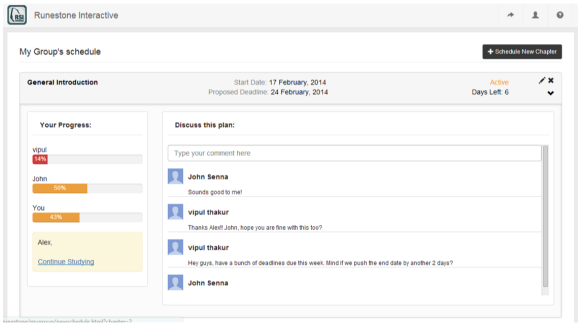

Figure 2: Screen capture from group support interface, showing discussion and progress bars

We used the interactive elements to create an *examples + practice* pattern in the book. We used the Active Code, Audio Tours, and Code Lens features to present *examples* to teachers. We followed the examples with *practice* using

Parsons Problems, multiple choice questions, or fill-in-the-blank questions. We interleaved examples plus practice as used in cognitive load studies [33] and in the pattern of one or two examples (better if contrasting examples) and one or two practice problems as recommended by Trafton and Reiser [34]. The practice sometimes asked participants to *change* code, but never to write code from a blank page. For example, one Active Code example computes the body mass index (BMI), given weight in pounds and height in inches. The multiple choice question that follows asks "Imagine that you are 5 foot 7 inches and weighed 140 pounds. What is your BMI?" The book contains a few short video snippets as examples, but in general, has less expository text or video than most books or ebooks. The design focus is on offering examples and encouraging practice.

Our design is informed by our studies of successful and unsuccessful computer science teachers. We know that successful computer science teachers rarely write code [15]. They have to read code, debug code, change code, and explain code. We designed our ebook to emphasize the knowledge and skills that high school computer science teachers really need.

One kind of expository text that we do include in the ebook are notes on pedagogical content knowledge (PCK) [18]. We explicitly tell teachers about the kinds of challenges students might encounter, how teachers might diagnose student misconceptions, and how to teach to address misconceptions and encourage better learning. Again, we were designing this ebook to serve the needs of teachers, and information on PCK is both something they need and a reason to read the ebook.

3. USABILITY STUDY

Our first question about our ebook was whether teachers judged it to be usable. We were concerned with the legibility and navigation usability, and also with the usability of the interactive elements. We decided to conduct a survey of teachers, asking them to judge the ebook on these elements.

We designed our usability survey based on a study of ebooks [7]. The authors in that study investigated and surveyed undergraduate students' design preferences for three different ebooks. They studied graphical user interface specifics, like page layout, font weight, and use of white space. We used their survey as a base, and extended it to include issues identified in other studies of ebooks [10] and to include the particular features that we were emphasizing in our ebook. The four interactive features that we asked our participants to rate were:

1. Active Code widget (Figure 3), which allows the user to edit and execute Python code, displaying any results or output, in the web browser.

2. Code Lens widget (Figure 4), which acts as a code visualization tool that allows the user to step through the code. It also displays variable values and program output

3. Parsons Problems widget (Figure 5), which allows the user to drag and drop blocks of code, from a bank of code blocks, into the correct order.

4. Multiple Choice widget, which asks the user a question about a coding concept, code, or output. It provides feedback explaining why the selected answer is correct or incorrect.

Figure 3: Example Active Code, with media computation [13] example

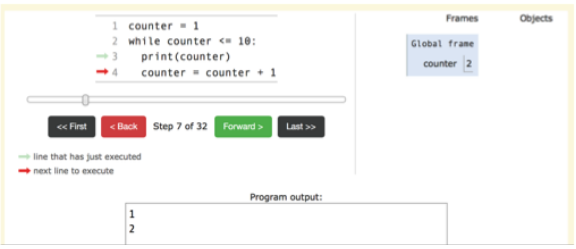

Figure 4: Example Code Visualization

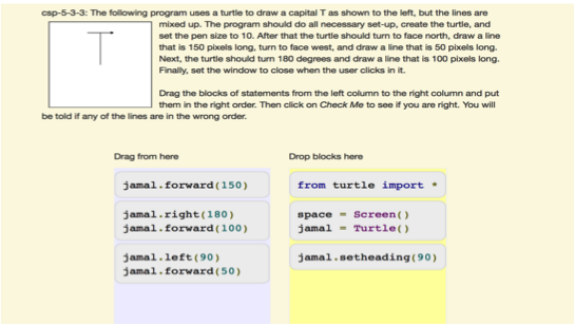

Figure 5: Example Parsons Problem

3.1 Method

After receiving human subject review clearance, we advertised for teacher participants in a computing education blog[1] and on CS teacher email lists. We gathered demographic data and only included survey responses from teachers with six months or more experience. All participants held at least a bachelor's degree and all had previously used some form of an ebook.

After the demographic data, all our participants completed a three-part survey comparing three different ebooks (like in the Chong et al. survey [7]). By comparing similar features on different platforms, we hoped to get more informed feedback that would highlight strengths and weaknesses. We had our participants compare a Runestone book to a Zyante ebook on Python[2] and to a CS Circles ebook[3]. Participants were offered a gift card for completing the study.

[1]http://computinged.wordpress.com
[2]https://zybooks.zyante.com/#/home
[3]http://cscircles.cemc.uwaterloo.ca/

We gauged the participants overall design preferences for the three ebooks as a whole using a five point Likert scale, where 1 is *Poor* and 5 is *Excellent*. They rated the three ebook designs on the following factors: navigation, page information, media arrangement, page layout, font, legibility, white space, and color. Additionally, they were asked to provide open-ended feedback about the design and usability of each ebook. For this section, the participants were provided with a URL and asked to rate the Runestone ebook first, then the Zyante ebook, and lastly the CS Circles ebook. In hindsight, we should have counter-balanced the order. Feedback on the user interfaces might be influenced by the order of presentation.

The next part determined the participants' usability and learnability preferences for the four interactive widgets found in each of the ebooks. Participants were solicited for feedback regarding the four widgets and their corresponding ebook platforms. For each widget-platform combination, a URL to a web page containing that specific widget implemented on the specific platform, Runestone, Zyante, or CS Circles, was provided to the participant. After interacting with the widget, they were asked to state the purpose of the widget. They were then asked to describe what they think each button and feature of the widget does. Finally, they were asked to report anything they found confusing or didn't particularly like about the widget.

The final part of the questionnaire asked the participants to report which platform implemented each widget the best. They compared the widgets on different platforms to one another, such that Code Lens on Runestone, Zyante, and CS Circles was compared against one another. After selecting their favorite widget-platform combination, the participants were asked to explain their reasoning for their selection.

3.2 Findings

For basic readability and navigation, Runestone and Zyante were similarly highly-ranked (Table 1). CS Circles was rated markedly less usable. Recall that 1 is *Poor* and 5 is *Excellent*.

	Runestone	Zyante	CS Circles
Navigation	3.09 (1.19)	3.68 (1.29)	2.50 (1.41)
Page Information	3.50 (0.80)	3.43 (1.12)	2.73 (1.12)
Media Arrangement	3.64 (1.00)	3.50 (0.96)	2.86 (1.08)
Page Layout	3.64 (0.95)	3.67 (0.97)	2.73 (1.12)
Font	3.68 (0.84)	3.68 (0.89)	3.59 (0.91)
Legibility	3.73 (0.70)	3.68 (0.78)	3.50 (1.19)
White Space	3.64 (0.90)	3.50 (1.19)	3.27 (1.20)
Color	3.95 (0.65)	3.73 (0.88)	3.38 (1.07)

Table 1: Readability and Navigation Results for ebooks, average (standard deviation) on a five-point Likert scale (1=poor, 5=excellent)

Participants were presented with four individual webpages each containing one of the widgets. They were asked to state what they believed the *purpose* of each widget was after interacting with it. In total, only three of the participants misidentified the purpose of any widget. Of those three, one participant's purpose response was deemed indeterminate for three of the four widgets, and one of those participants could not determine the purpose for the Active Code, Code Lens, or Parsons Problem widgets.

Participants were asked to select which platform had their favorite implementation for each of the four interactive widgets. For this selection, eighteen responses, one from each participant, were given. On average, participants ranked all the Runestone design factors as superior (Table 2).

	Runestone	Zyante	CS Circles
Active Code	10	6	2
Code Lens	7	8	3
Parsons Problem	12	5	0
Multiple Choice	10	6	1

Table 2: Number of votes as "best" for each widget by platform

Not all of the results were stellar. One participant told us that "They were all abysmal interfaces that would cause me to reject the book violently." The other 17 participants liked the ebooks, with Runestone receiving the greatest support. Overall, the evidence supported the belief that our ebook would not get in the learners' way.

4. USAGE STUDY

Our usage study asked teachers to read the first eight chapters of our ebook at their own pace. These chapters cover introductory computing concepts in Python, such as naming variables and repeating code with while and for loops. We asked our study participants to take a pre-test of their knowledge of introductory CS. The pre-test was composed mostly of questions from other studies [31, 22], so that we measured general introductory computer science knowledge. We asked our participants to take a post-test after every two chapters, where each post-test was designed to test the content from those chapters.

4.1 Method and Participants

After receiving permission from the human-subjects review board at Georgia Tech, we advertised through the same blog and email lists for teachers that we had used for our usability study. Our initial recruitment email included the purpose of the research and the study criteria. Participants were required to have had two months or *less* of prior coding experience and knowledge. Participants who responded to the recruitment solicitation were provided with a link to the pretest.

We only accepted participants who scored less than 40% on the pre-test. They were emailed a web link to the ebook and the four post-tests. They were instructed to read and interact with the chapters of the ebook at their own pace. Upon reading two chapters of the ebook, they were then to complete the corresponding post-test. While completing the post-test, the participant was instructed not to refer back to the ebook or use any outside material that might assist them in answering the questions. Teachers who completed all of the post-tests were offered a $50 USD gift card.

Ten teachers qualified for the study based on their pre-test scores. Of those ten, seven completed at least one post-test, and five completed all four post-tests. However, the log shows that one of the teachers, Jim, completed all four post-tests but only actually completed chapters one through six in the ebook. Another teacher, Debra, completed all eight chapters of the ebook, but didn't take any of the post-tests. It's unclear how to categorize these results

by MOOC research terms, since completing a pre-test is already more commitment than many "Unsure" or "Browsing" MOOC learners [30]. Our ten participants probably best reflect MOOC participants who complete the first homework. Having five (50%) teachers out of 10 complete the eight chapters of the ebook is still quite high, though our participants were compensated with a gift card rather than simply receiving a certificate. When considering the economics of professional learning for teachers, $50 is a small cost, but it probably had a significant impact on completion rates. The logs show that Gina only did five of the 17 possible interactive activities in chapter 8 and Vicki did only seven, so they may have been focusing on "completing" the chapter to earn the gift card. Still the completion rate in our compensated study was higher than is typically seen with MOOCs and very high compared to CS teacher MOOCs.

It's worth considering the teachers who did not take any post-tests:

- One teacher (a white female) visited the ebook on three different days, but her only interaction with the ebook was running one example.
- A second teacher (a white female) completed the first two chapters and started chapter 3, but did not take any post-tests.
- A third teacher (a black female) actually completed all eight chapters in the study, and four more, all within the first three days of the study. But she did not take any post-tests.

We can compare these to the categories developed for MOOC participants by Reich [30]. The first case sounds like the "unsure" learner. The latter two may have been "browsers." They may have found the information that they wanted, but chose not to "complete" or "participate" (as in taking the post-tests). Compensation likely had a significant impact on our completion rate, but did not eliminate non-completers.

4.2 Pre-Test and Post-Test Results

Table 3 summarizes the demographics and scores for the seven teacher participants who completed at least one post-test. Four of the participants were female. Two were self-reported members of under-represented minority groups. (One did not report race/ethnic group.) Their self-reported programming knowledge was "Average" for two participants, and below average for the rest.

None of the seven teachers scored more than three questions right on the nine question pre-test. All but one of the participants scored above 50% on the first two post-tests. Only five participants completed the third post-test, and only one scored less than 25%. Two participants of the five completers scored below 50% on the last test. However, these were the participants, Gina and Vicki, who utilized much less of the interactive features.

Table 3 may seem to suggest that learning occurred from pre-test to post-test, but while the participants had low-scores on the pre-test and higher scores on the post-test, the tests were not close matches. The pre-test was considerably harder than any post-test, given that it was drawn from research instruments measuring introductory course knowledge. For example, one of questions from the pre-test appears in Figure 6, which tests knowledge of lists. None of the first eight chapters focused on lists. By saying that all the participants scored less than 40% on the pre-test, we are saying that the participants knew less than 40% of introductory course knowledge. A few of the post-test questions were fairly easy, e.g., "The kind of data which can be letters, digits, and other characters, usually delimited by quotes is a (a) Integer, (b) Float, (c) String, or (d) Double." Some of the post-test questions were difficult as shown in Figure 7. Only two of the five teachers who took this post-test got this question correct.

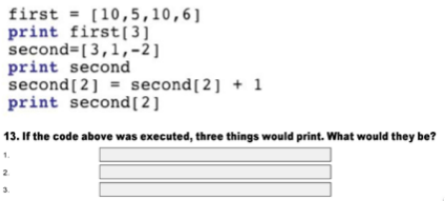

Figure 6: One of the pre-test questions

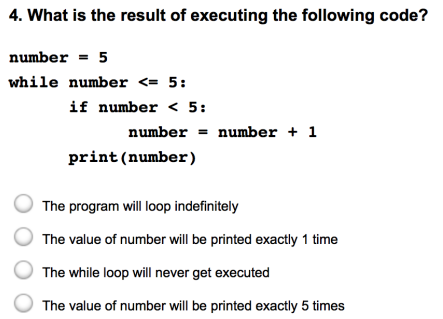

Figure 7: One of the post-test questions

The teacher's open-ended comments show us that most of the teachers appreciated the features of the ebook.

- "I thought the drag and drop code (Parsons problem) was very good. I thought the sequence and the scaffolding of information was good. I liked that you could type or change different codes into the boxes and it would produce different results. I liked that it was dynamic enough to be able to give you something in return and you could see the results (turtle triangle)."
- "I feel like this would be an effective and beneficial tool for students and teachers."
- "It's fun to modify the example programs."

Teacher confidence is a critical factor in developing a sense of identity and improving as a teacher [27, 28]. Most teachers self-reported that they were ready to teach using the content after using the ebook, claiming that their ability was "Average" or even "Excellent." One of the teachers, Gina, rated her ability to teach this content as "below average" and another, Vicki, rather her ability to teach the content as "extremely poor." Interestingly the teachers who reported higher confidence in their ability to teach the material spent much more time working through the ebook and used more of the interactive features than teachers who didn't feel confident.

Pseudonym	Gender	Race	Programming Knowledge	Pre-test (out of 9)	Ch1-2 (out of 3)	Ch3-4 (out of 3)	Ch 5-6 (out of 4)	Ch 7-8 (out of 3)
Mary	Female	Black	Average	22%	66%			
Vicki	Female	Black	Extremely Poor	0%	33%	33%	25%	0%
Fred	Male	N/A	None	33%	66%	100%	50%	66%
Gina	Female	White	Below Average	22%	66%	100%	75%	33%
Jim	Male	White	Below Average	11%	66%	100%	75%	66%
Kay	Female	White	Average	33%	100%			
Cabe	Male	White	Extremely Poor	33%	100%	100%	100%	66%

Table 3: Demographic, pre-test, and post-test data for participants who completed at least one post-test

> "Need more exercises actually writing codes that will do something. I can understand what it says but I don't know that I would be able to begin to start my own still."

Several teachers commented that they needed more practice. In response to the question, "Do you feel like this would be an effective and beneficial tool for students? teachers?", one participant responded:

> "Absolutely. The interaction and ability to immediately observe how modifications change the output is quite helpful. Of course, I would benefit from a LOT more practice exercises to accompany the text. I feel like I get it, but won't hold on to it otherwise."

It is not surprising that teachers want more practice since the first eight chapters only cover variables and loops. It will be interesting to see what teachers report after completing all 20 chapters in the current version of the ebook. These additional chapters include three more chapters on loops as well as chapters on conditionals, lists, and working with data.

4.3 Log File Analysis of Use

We studied the log files, which track events in the ebook (like clicking the "Run" button in Active Code), as a source of data on what teachers did in the ebooks. Table 4 summarizes the use of the ebook interactive features. Each unique widget (e.g., one program in an Active Code, one Parsons Problem, one visualization in the Code Lens) is counted once in the table to give a sense of the kind of use that participants made of the interactive features in the ebook. The table includes the last chapter visited and performance on the last post-test to give a sense of the resultant performance. (All of the post-test scores are available in Table 3.) For example, Vicki used the fewest of the interactive features, and she had the lowest performance on the final post-test. Kay gave up before the others, and she was the only participant to never attempt any Parsons Problems. To give a sense of perspective on these, there were 15 Parsons Problems in the eight study chapters and 40 multiple-choice questions. Fred had far more than that – by going beyond Chapter 8. He skipped around and visited other chapters up to Chapter 11 during the study period.

The teachers who had the highest scores on the final post-test also made the greatest use of the interactive features. Fred, Jim, and Cabe used the most Parsons Problems, the most Active Code, and the most Code Lens. The teachers who did not make it as far into the book or who scored the lowest on the final post-test did not use the interactive features nearly as much. The counts in Table 4 do not show how *much* use is made of each of these problems. For example, Fred interacted with 23 Parsons Problems, but checked his problems a total of 59 times, suggesting that he struggled with getting some of them right. Jim interacted with 9 Parsons Problems, with 39 attempts. Vicki made 9 attempts on 5 problems.

All participants answered the multiple choice and fill-in-the-blank questions. In fact Vicki and Gina ignored most of the other interactive features by the last chapter, but did attempt the multiple choice questions. Vicki answered more multiple-choice questions than Jim, but she rarely got any right and only once changed her answer. There is no obvious relationship between the amount of multiple choice and fill-in-the-blank questions attempted and better performance on any post-test. With more participants and/or with more post-test questions, we might be able to find an interaction with other components. It may be that use of the worked examples in Active Code and Code Lens *plus* practice with multiple choice and fill-in-the-blank questions leads to more learning.

Another distinction between the participants was on use of the Code Lens. Vicki interacted with the Code Lens widget only once, while Fred interacted with 30 Code Lens problems. More telling still is the number of interactions with each of those Code Lens examples. Vicki only viewed a Code Lens on an Active Code example one time, but didn't actually step through the execution of the code using the Code Lens. Jim explored the Code Lens visualizations (e.g., stepping through the program) 55 times over his 5 problems. Fred explored the visualizations with 318 steps over his 5 problems.

Participants did interact with the code (using the Active Code widget). Table 5 disaggregates *how* they interacted with code. We see much more execution of code than editing, but we do see a significant amount of code editing – especially from the participants who performed the best on post-tests. Vicki never edited any examples and performed the worst on the post-tests. What is probably even more important for learning is the number of syntax errors that they encountered, as an indication of exploration and potentially "desirable difficulties" [4]. It was highest for Fred, who is one of the teachers who got the best score (66%) on the last post-test. The amount of interaction with code is interesting since our book *encouraged* interaction with code, but did not *require* it. Teachers were never required to code from a blank editing screen, instead they were asked to modify the examples.

We did see use of the Audio Tours, though they didn't listen to every Audio Tour. Fred used them the most and did perform the best overall, but Fred also did *many* other

Pseudonym	Parsons Problems	Active Code	Code Lens	Audio Tours	Multiple Choice	Fill in the Blank	Syntax Errors	*Last Chapter Visited*	*Last Test (Score)*
Mary	3	13	1	0	19	1	3	5	Ch1-2 (66%)
Vicki	5	17	1	5	40	0	0	8	Ch7-8 (0%)
Fred	21	67	30	26	60	2	12	8	Ch7-8 (66%)
Gina	6	3	2	4	43	0	0	8	Ch7-8 (33%)
Jim	9	23	8	2	24	1	2	7	Ch7-8 (66%)
Kay	0	7	2	1	7	1	0	3	Ch1-2 (100%)
Cabe	15	35	9	3	45	1	2	8	Ch7-8 (66%)

Table 4: Counts of use of unique interactive problems that participants attempted, by type

	Edited Code	Ran Code	Syntax Errors
Mary	20	36	3
Vicki	0	19	0
Fred	93	153	12
Gina	1	4	0
Jim	14	42	2
Kay	3	12	0
Cabe	18	52	2

Table 5: How participants used the coding problems

	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8
Mary	0:47	0:09	0:33	0:06	0:23			
Vicki	0:16	0:14	0:17	0:06	0:09	0:16	0:09	0:03
Fred	0:24	0:39	1:41	0:57	0:44	1:46	1:35	2:24
Gina	0:43	0:39	0:36	0:12	1:48	0:29	0:20	0:10
Jim	0:21	0:33	0:57	0:14	0:26	0:16		
Kay	1:10	0:25	0:15					
Cabe	0:16	0:07	0:32	0:08	0:27	0:32	0:42	0:30

Table 6: Time spent in each chapter by participant

activities.

We do note that nearly all of the participants who made the most use of the ebook were all male. Does our ebook somehow bias engagement with the activities? We can't tell from these results with a small number of participants, but it's something we will watch as we scale up use with the complete ebook. Analysis of the one female, Debra, who finished eight chapters, but didn't take any of the post-tests does show similar use of the features as the three males who used the most features. We also hope that the built in support for small reading groups ("book clubs") will make working through the ebook more social and engaging.

Overall, the results support the hypothesis that more interaction with the book was correlated with better results on the post-tests. Cabe is a great example of use leading to learning. Note that he rated his knowledge of programming as "Extremely poor" on the pre-test. He had the most interaction with the book after Fred. His scores on the post-tests were good. The results support our hypothesis that the the design of the ebook is supporting teacher learning. How much learning occurred was most likely a function of how much effort the learners made in exploring, trying out the features, and making mistakes.

4.4 How did they spend their time?

Recall that our design rationale for the ebook was to offer a predictable model that could fit into the teachers' lives. We imagined them reading the book and doing the interactive elements in small bits of time. Table 6 lists our estimate of the amount of time each participant spent in each chapter. Since our log file only tracks *events*, we can only report on the time between the first event in a chapter and the last event in a chapter. We cannot account for time spent reading, for example.

There's an enormous variance between the participants. The participants who did the best on the post-tests also spent more time in the chapters and in the ebook in total. Time on task leads to more learning, so that correlation is not surprising. There are trends suggesting the variance between time needed to complete each chapter. Chapter 3 took more time than Chapters 2 or 4 for most participants. This is not surprising as Chapter 2 only has some text, a few pictures, and two multiple-choice questions. Chapter 4 has only 10 interactive features while Chapter 3 has 31.

Tracking individuals through the log files provides insight into how teachers read the ebook. We do see teachers fitting the ebook into bits of time. We see Vicki starting the book at 3:19 pm on one day and ending that session at 3:35, and the next day, she starts at 4:13 pm and ends at 4:24 pm. Kay reads for a 70 minutes starting at 3:30 one day, and for 25 minutes at 5:51 pm the next day. Teachers may not always have been able to predict how much time a chapter would take. Fred's sessions with the ebook went past midnight on several occasions. In general, teachers tended to complete a chapter within a single session, though some did work on the same chapter on two different days. Since some of the chapters were considerably longer than others, the time per chapter is less predictable than we would have liked. We might fit reading the ebook better into the small chunks of time in their lives if we had more, smaller chapters, so that the chapter of a Runestone ebook was closer to the sense of pages in a physical book. It might also be helpful to add an estimate of the time it will take to complete each chapter based on this study.

Most teachers worked though the interactive features in the order that they appear in the chapters. Gina and Vicki both skipped more than half of interactive features in Chapters 5 and 8 as shown in Table 7 and Table 8. The tables show the order that each teacher used each feature. For the Active Code feature the 'r' means run and the 'e' means edit. Numbers in parentheses mean the number of times each thing occurred, for example (11)r means running the code 11 times. For the Multiple Choice feature the 'c' means the answer was correct and the 'w' means it was wrong. For the Parsons Problems we list the number of tries it took to get the correct answer. For the Code Lens feature the 'f'

means forward.

Activity	Vicki	Gina	Fred	Cabe	Jim
Page		1, 3	1	1	2
Active Code	1r		2r	2rr	1r, 7r
Audio Tour			3		
Active Code	2r		4rrr, 6re	3r, 5er	3r, 5er, 8r
Audio Tour					
Mult. Choice	3c	4c	5c	4c	4c
Active Code	4r		7r	7r	
Audio Tour					
Mult. Choice	5w	5 – 2 tries	8c	6c	6w,9c
Parsons		6n	9 – 1 try	7 – 2 tries	10 – 1 try
Parsons	6w	2 – 11 tries	10 – 2 tries	8 – 3 tries	11 – 2 tries
Active Code	7r		(11)r (7)e	9r	12r
Audio Tour					
Parsons			12 – 2 tries	10 – 1 try	13 – 5 tries
Active Code			13r, 15re	11r, 13re	14r
Audio			14		
Mult. Choice	8w	7c	16c	12c	15c
Active Code	9r		17 (6)r(7)e	14 rerre	16 (5)r(5)e
Mult. Choice	10c	8c	18c	15c	17– 2 tries
Parsons			19 – 1 try	16 – 2 tries	18 – 2 tries
Video	11	9	20		

Table 7: Order of feature use in Chapter 5

We also see evidence of teachers using multiple interactive features repeatedly which may indicate that they are trying to understand a concept. For example we see several examples of cycles within the examples+practice structure. In Chapter 5 we see Fred run an Active Code example three times as the fourth thing he does and then answer a multiple-choice question correctly and then go back and run and edit the Active Code example again. Cabe and Jim do the same thing. In chapter 8 Fred comes back to the same Active Code example three times. In between he uses the Code Lens and answers a Multiple Choice question. This type of deeper exploration should improve learning.

To understand how to help the learners who are less successful, we looked in some detail at the log file for Vicki. From just the log file, we cannot be certain about intentionality. We can note when activity was different than what we expected.

- When Vicki started the first chapter, her first practice activity was to answer a multiple choice question about an example program. She got the answer wrong, and then went back to run the program and listened to the Audio Tour. Using the example to help explain a multiple choice question result is just what we wanted, but we were somewhat surprised that a teacher would try to answer the question first before actually studying the example.

Activity	Vicki	Gina	Fred	Cabe	Debra
Page			1	1	1
Code Lens			2 (32)f	2 (32)f	2 (32)f
Mult. Choice	1c	1 – 2 tries	3 – 2 tries	3 – 2 tries	3 – 2 tries
Mult. Choice	2c	2c	4y	4y	4 – 2 tries
Parsons	4w		5 – 1 try	5 – 1 try	
Parsons	3w		6 – 1 try	6 – 1 try	5 – 9 tries
Code Lens			7 - (19)f	7 - (19)f	6 (2)f, 8 (17)f
Mult. Choice	5c	3 – 3 tries		8c	7 – 2 wrong tries, 9y
Code Lens				9 - last	10 (78) fwd
Video			9	10	
Video			10	11	
Active Code			12r, 14e, 18eer	12r	11r
Code Lens			13		
Audio			11(6)		
Mult. Choice	6w	4 – 6 tries	15c	13c	12 – 3 tries
Video			16	14	
Mult. Choice	7w	5 – 2 tries	17 – 2 tries	15c	13 – 2 tries

Table 8: Order of feature use in Chapter 8

- Vicki only changed one of her answers on a multiple choice question. Because it only happened once, it may have been a mistake (a stray mouse click) or she didn't notice. Teachers may not realize that they can answer the multiple choice questions more than once until they select the correct answer.

- Vicki got most of the multiple-choice questions that she answered wrong (38 wrong vs 10 correct). Most of the participants answered the multiple-choice questions until they got them correct, which probably led to more learning.

- Vicki did click on the Parsons Problems *Check* button, to see if the answer was correct. But it was always wrong, because she never once dragged any of the code blocks into position! Though she did watch one video, she did not watch the video on how to do Parsons Problems. She doesn't seem to have figured out how to use that widget.

- Vicki ran some of the Active Code examples in chapters 1-6, but never attempted to edit any of them, even though some of the multiple choice questions asked her to in order to answer the question.

- In chapters 7 and 8 Vicki only answered the multiple-choice questions and clicked the *Check* button on the Parsons problems without dragging any code blocks to the solution area. It seems that she rushed through the ebook, perhaps to finish in time to get the $50 gift card for "completing".

5. CONCLUSION

This paper presents the results of two studies of our ebook designed for high school teachers learning CS Principles. We investigated two questions with these studies:

- Did the teachers find the ebook to be *usable* so that their time is spent on fruitful activities, not on figuring out an interface?
- Did the teachers find the ebook to be *efficient* so that the learning benefit seemed worthwhile for the time spent? We measured that latter question in terms of completion rates, pre-test/post-test comparisons, and log file analyses of activities.

From our usability survey, the ebook was found to be legible, easy to navigate, and usable. The interactive features were as good as if not better than comparable features in other ebooks. Our usability participants did not find it difficult to use the ebook. However, our usage study teachers (who did have less experience with computing than those who partcipated in the design study) did occasionaly struggle with the interface, such as not attempting to solve any of the Parsons problems.

The completion rate was 5 of 10 (50%), with 7 of 10 (70%) completing at least one post-test. This result suggests that about half the teachers found the activity worthwhile. That is a higher completion rate than we might expect from MOOC results. However, our results come from a small study, and the teachers were compensated. We cannot be sure that we have better completion rates than a traditional MOOC until we scale up use of the ebook and do not offer compensation.

Teachers generally valued the ebook, but found it to be too short, which is not surprising given that they only completed eight of the 20 chapters. Our pre-test and post-tests did not measure exactly the same knowledge, but more use of the interactive features did appear to lead to higher post-test scores. The log file analysis suggests changes that might improve use. We need more and smaller chapters. We cannot be sure yet about the value of each of the widgets. Scaling up use may give us more data to determine the effectiveness of individual components.

We believe that the best use of the ebook is in a blended format. We would like to introduce the ebook to teachers in a face-to-face professional development setting. This should help the teachers develop social bonds that could continue into a small group "book club" setting. This could provide social pressure to complete the ebook. The usability problems that Vicki had with the ebook might be reduced if teachers had the ebook features introduced in a face-to-face setting.

The critical finding in this paper is that our ebook approach is on a productive path. Teachers are finding our ebook *usable* and *effective*. There are complaints that we have not yet provided *enough* practice. Further, we have not yet tested all of our features, such as our groupwork support. The results thus far support the belief that this approach may provide valuable online learning opportunities for inexperienced computing teachers.

6. ACKNOWLEDGEMENTS

Our thanks to the teachers who participated in our study. This paper is based on work supported by the National Science Foundation under Grant 1138378. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] O. Astrachan, A. Briggs, L. Diaz, and R. B. Osborne. CS principles: development and evolution of a course and a community. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 635–636, New York, NY, USA, 2013. ACM.

[2] O. Astrachan, R. Morelli, D. Barnette, J. Gray, C. Uche, B. Cowles, and R. Dovi. CS principles: piloting a national course. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 319–320, New York, NY, USA, 2012. ACM.

[3] K. Benda, A. Bruckman, and M. Guzdial. When life and learning do not fit: Challenges of workload and communication in introductory computer science online. *ACM Transactions on Computing Education*, 12(4):1–38, 2012.

[4] E. L. Bjork and R. A. Bjork. Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. *Psychology and the real world: Essays illustrating fundamental contributions to society*, pages 56–64, 2011.

[5] N. C. C. Brown, M. Kölling, T. Crick, S. Peyton Jones, S. Humphreys, and S. Sentance. Bringing computer science back into schools: lessons from the uk. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 269–274, New York, NY, USA, 2013. ACM.

[6] M. E. Caspersen and P. Nowack. Computational thinking and practice: A generic approach to computing in Danish high schools. In A. Carbone and J. Whalley, editors, *The 15th Australasian Computer Education Conference (ACE 2013)*, Adelaide, South Australia, February 2013. Conferences in Research and Practice in Information Technology (CRPIT).

[7] P. Chong, Y. Lim, and S. Ling. On the design preferences for ebooks. *IETE Technical Review (Medknow Publications & Media Pvt. Ltd.)*, 26(3), 2009.

[8] B. Dasarathy, K. Sullivan, D. C. Schmidt, D. H. Fisher, and A. Porter. The past, present, and future of MOOCs and their relevance to software engineering. In *Proceedings of the on Future of Software Engineering*, FOSE 2014, pages 212–224, New York, NY, USA, 2014. ACM.

[9] A. D. Edgcomb and F. Vahid. Effectiveness of online textbooks vs. interactive web-native content. In *121st ASEE Annual Conference and Exposition*, number Paper ID #10004, 2014.

[10] J. B. Fenwick, Jr., B. L. Kurtz, P. Meznar, R. Phillips, and A. Weidner. Developing a highly interactive Ebook for CS instruction. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 135–140, New York, NY, USA, 2013. ACM.

[11] P. J. Guo. Online Python Tutor: Embeddable

web-based program visualization for CS education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. ACM.

[12] P. J. Guo, J. Kim, and R. Rubin. How video production affects student engagement: An empirical study of MOOC videos. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 41–50, New York, NY, USA, 2014. ACM.

[13] M. Guzdial. Exploring hypotheses about media computation. In *ICER '13: Proceedings of the ninth annual international ACM conference on International computing education research*, pages 19–26, New York, NY, USA, 2013. ACM.

[14] M. Guzdial. Meeting student and teacher needs in computing education. *Commun. ACM*, 57(12):10–11, Nov. 2014.

[15] M. Guzdial. Preparing teachers is different than preparing software developers: [wipsce'14 keynote]. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, WiPSCE '14, pages 1–1, New York, NY, USA, 2014. ACM.

[16] M. E. Hansen, A. Lumsdaine, and R. L. Goldstone. An experiment on the cognitive complexity of code. In *Proceedings of the Thirty-Fifth Annual Conference of the Cognitive Science Society*, Berlin, Germany, 2013. Cognitive Science Society.

[17] J. Helminen, P. Ihantola, V. Karavirta, and L. Malmi. How do students solve parsons programming problems?: An analysis of interaction traces. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, pages 119–126, New York, NY, USA, 2012. ACM.

[18] P. Hubwieser, J. Magenheim, A. Mühling, and A. Ruf. Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 1–8, New York, NY, USA, 2013. ACM.

[19] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 73–84, New York, NY, USA, 2006. ACM.

[20] D. Koller, A. Ng, C. Do, and Z. Chen. Retention and intention in massive open online courses: In depth. *Educause Review*, 2013.

[21] A. Korhonen, T. Naps, C. Boisvert, P. Crescenzi, V. Karavirta, L. Mannila, B. Miller, B. Morrison, S. H. Rodger, R. Ross, and C. A. Shaffer. Requirements and design strategies for open source interactive computer science eBooks. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports*, ITiCSE -WGR '13, pages 53–72, New York, NY, USA, 2013. ACM.

[22] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Moström, K. Sanders, O. Seppälä, B. Simon, and L. Thomas. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.*, 36:119–150, June 2004.

[23] R. E. Mayer and R. Moreno. A split-attention effect in multimedia learning: Evidence for dual processing systems in working memory. *Journal of Educational Psychology*, 90:312–320, 1998. Modality Principle.

[24] B. Miller and D. Ranum. Runestone interactive: Tools for creating interactive course materials. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 213–214, New York, NY, USA, 2014. ACM.

[25] B. B. Morrison, L. Ni, and M. Guzdial. 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*, ICER '12, pages 47–54, New York, NY, USA, 2012. ACM.

[26] S. Y. Mousavi, R. Low, and J. Sweller. Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology*, 87(2):319–334, 1995.

[27] L. Ni. *Building professional identity as computer science teachers: Supporting high school computer science teachers through reflection and community building*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, December 2011.

[28] L. Ni and M. Guzdial. Prepare and support computer science (CS) teachers: Understanding CS teachers' professional identity, 2011.

[29] D. Parsons and P. Haden. Parson's programming puzzles: A fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, pages 157–163, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[30] J. Reich. MOOC completion and retention in the context of student intent. *Educause Review*, 2014.

[31] J. Sorva. *Visual Program Simulation in Introductory Programming Education*. Doctor of science in technology, Aalto University School of Science, 2012.

[32] C. Spradling, D. Linville, M. P. Rogers, and J. Clark. Are MOOCs an appropriate pedagogy for training K-12 teachers computer science concepts? *Journal of Computer Science in Colleges*, 30(5):115–125, May 2015.

[33] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285, 1988. Theory behind worked examples.

[34] J. G. Trafton and B. J. Reiser. *The contributions of studying examples and solving problems to skill acquisition*, pages 1017–1022. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1993.