Runestone: A Platform for Free, On-line, and Interactive Ebooks

Barbara J. Ericson
barbarer@umich.edu
University of Michigan
School of Information
Ann Arbor, Michigan

Bradley N. Miller
brad@runestoneinteractive.com
Luther college / Founder Runestone Interactive
Minneapolis, Minnesota

ABSTRACT

The Runestone platform is open-source, extensible, and serves free ebooks to over 25,000 learners a day from around the world. The site hosts 18 ebooks for computing courses. Some of these ebook have been translated into several languages. There are ebooks for secondary computer science (AP CSP and AP CSA), CS1, CS2, data science, and web programming courses. The platform currently supports executable and editable examples in Python, Java, C, C++, HTML, JavaScript, Processing, and SQL. Runestone provides features for instructors, learners, authors, and researchers.

Instructors can create a custom course from any of the existing ebooks and their students can register for that course. Instructors can create assignments from the existing material or author new problems, grade assignments, and visualize student progress. Learners can execute and modify examples and answer practice questions with immediate feedback. Runestone includes common practice types, such as multiple-choice questions, as well as some unique types, such as adaptive Parsons problems. Authors can modify the existing ebooks or write new ebooks using `restructuredText`: a markup language. Researchers can create and test new interactive features, run experiments, and analyze log file data.

This paper describes the architecture of the platform, highlights some of the unique features, provides an overview of how instructors use the platform, summarizes the research studies conducted on the platform, and describes plans for future development.

CCS CONCEPTS

• **Social and professional topics** → **Informal education; Student assessment; K-12 education; Adult education.**

KEYWORDS

on-line learning, ebooks, adaptive learning, practice tools, intelligent ebooks, Parsons problems

ACM Reference Format:

Barbara J. Ericson and Bradley N. Miller. 2020. Runestone: A Platform for Free, On-line, and Interactive Ebooks. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, March 11–14, 2020, Portland, OR, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3328778.3366950>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-6793-6/20/03...\$15.00

<https://doi.org/10.1145/3328778.3366950>

1 INTRODUCTION

Early electronic books were simply digital copies of paper books [5], providing portability and searchability [48]. However, current web-based ebook platforms for computer science courses, such as the open-source OpenDSA platform [19, 42, 43], the commercial zyBook platform, and the opensource Runestone platform [34, 35] offer much more. They include instructional material and a variety of practice problems with immediate feedback, which is crucial for efficient learning [4]. An ITiCSE working group predicted that traditional CS textbooks will likely be replaced by online materials, such as ebooks, that integrate content and assessment [28].

For example, commercial zyBooks include minimal text, images, code animation, multiple-choice questions, fill-in-the-blank questions, mixed-up code (Parsons) problems [38], matching problems, and executable code with unit tests. Studies of interactive zyBooks versus traditional textbooks with static content have provided evidence that interactive ebooks improve student performance, engagement, and the amount of time students spent in the ebook [10–12]. Another recent study by a research group at Aalto University using their own platform also found that an interactive ebook improved student motivation and learning compared to a static ebook [39].

Brad Miller started creating the Runestone ebook platform in 2011. He wanted to have ebooks with embedded code that the learner could run and edit. Brad also wanted ebooks to be free so that everyone has the chance to learn computer science, even if they can not afford a \$200 textbook. Brad's goal was “*democratizing textbooks for the 21st century*”.

2 ARCHITECTURE

Authors write Runestone ebooks using `restructuredText` (rst) [20], which is a minimal markup language that is similar to Markdown [22]. The source for the ebook is converted from `restructuredText` into HTML5 by the “runestone build” command as shown in Figure 1. The build process uses Sphinx [3]; a tool written in Python that creates documentation. Sphinx contains many built-in *directives* for things like figures and notes, but is also extensible. Runestone developers have added new directives to Sphinx to create new interactive features for the ebooks.

For example, the *activecode* directive shown in Figure 2 is used to embed executable and editable code. The displayed code is shown in Figure 3 and the output is shown in 4. If the code is Python, it is run as JavaScript on the client using Skulpt [21]. If the code is JavaScript or HTML it is run directly in the browser. If the code is Java, Processing, C, C++ or SQL the code is executed on the Runestone server, which is built on top of the web2py [9] application

Figure 1: The Runestone architecture.

framework. It provides web services and supports the instructor tools.

The build process produces either a static website, or a set of HTML5 templates that can be used to serve hundreds of custom courses as shown in Figure 1. The Runestone ebook server combines the templates for each page with information from the database about the student and the course, and then serves each student a custom page. As the student interacts with the various components on the page, an API is used to gather this data and store it in the database.

```
.. activecode:: sql1
  :language: sql
  :autograde: unittest
  :dburl: https://runestone.academy/overview/_static/bikeshare.db

  select bike_number, max(duration)
  from trip_data
  group by bike_number
  order by max(duration) desc
  limit 4;

  ====
  assert 0,0 == W00379
  assert 0,1 == 86355
  assert 2,1 == 86336
```

Figure 2: The source for an activecode directive that embeds executable SQL in the ebook.

Figure 3: The displayed SQL problem.

Figure 4: The output from the unit tests and SQL.

3 RUNESTONE FEATURES

The first major version of Runestone supported text, images, videos, executable and editable Python code, and a Python code visualizer/stepper (Python Tutor) [23, 35]. The next major version included multiple-choice questions with feedback based on the selected answer, fill-in-the-blank questions, mixed-up code problems (Parsons problems), embedded questions during code stepping, and discussion boxes to allow for group discussion on a problem [34]. The current version includes common interactive features such as short-answer and matching questions, support for pair programming, as well as several unique features: audio tours of code, clickable-code, adaptive Parsons problems, timed exams, and a spaced, interleaved, and retrieval-based practice tool. See [32] for an example for each Runestone feature as well as the restructuredText that generated that example.

3.1 Audio Tours of Code

The *activecode* directive is used to add executable and editable code as shown in Figure 3. It can include up to five optional audio tours. In an audio tour, one or more lines of code are highlighted as the audio plays as shown in Figure 5. This feature takes advantage of the fact that humans have dual channels which allow simultaneous processing of visual and audio information [30, 31].

3.2 Clickable Code

In clickable code the user clicks on one or more lines of code to answer a question. Each line that the user clicks is highlighted in yellow. The user clicks the *Check Me* button to submit an answer. The correct lines that were clicked remain highlighted in yellow and any incorrect lines that were clicked are boxed in red as shown in Figure 6. Textual feedback is displayed under the question.

Figure 5: An audio tour highlights one or more lines of codes as the audio plays that explains those lines.

Figure 6: A clickable code question that shows the correct answers still highlighted in yellow and the incorrect answers boxed in red.

3.3 Adaptive Parsons Problems

While several ebook platforms have mixed-up code problems (Parsons problems) [38], only Runestone has adaptive Parsons problems. In a Parsons problem the learner places the code in the correct order to solve a problem as shown in Figure 7. In an adaptive Parsons problem the difficulty of a problem is based on the learner's performance. Parsons problems can be added using the *parsons* directive.

Parsons problems can display labels to support collaborative work as shown in Figure 8. These labels are also useful when Parsons problems are used in paper-based exams [6, 8]. On a paper-based exam the learner can simply write the block labels in the correct order. This reduces the cognitive load for test takers since they do not have to generate a solution. Paper-based Parsons problems are faster to grade than the equivalent code writing problem and the grading is more consistent [6]. Scores on Parsons problems have correlated with scores on code writing problems [6, 8].

Figure 7: An example Parsons problem in Python. The mixed-up blocks are shown on the left and the correct solution is shown on the right.

Parsons problems can also contain *distractors*, which are extra code blocks that are not needed in a correct solution. Distractors can either be randomly mixed in with the correct code or shown paired with the corresponding correct code as shown in Figure 8. Some Parsons problems also require the learner to indent the code correctly. These are called two-dimensional Parsons problems. Some languages, like Python, require indentation to specify the code structure, such as which statements are in the body of a loop.

Ericson invented two types of adaptation for Parsons problems [15]. In *intra-problem adaptation* if the learner is struggling to solve the current problem it can dynamically be made easier by disabling distractor blocks, providing indentation, or combining blocks. In *inter-problem adaptation* if the learner solved the last problem in just one attempt the next problem can be made harder by randomly mixing all distractors in with the correct code blocks. If the learner took many attempts to solve the last problem the next problem can be made easier by pairing the correct and distractor code or removing distractors.

Figure 8: An example Parsons problem in Python. Notice that there are pairs of correct and incorrect (distractor) code blocks on the left side with an "or" on the left side of the pair of blocks. The "or" indicates that the learner should use only one of the two blocks in a solution.

3.4 Timed exams

Authors can add timed exams to ebooks using the *timed* directive. Instructors can also create timed exams from a web based interface. These are intended for summative assessment and can contain any of the practice question types such as multiple-choice questions, fill-in-the-blank questions, and Parsons problems. A timed exam can have a time limit or can just display the amount of time the user has spent on the exam. The learner must click the Start button to start a timed exam and then one question is displayed at a time as shown in figure Figure 9. The user can use the question number buttons to jump to a particular question.

24-2-8: The Traveling Salesman Problem is a problem in theoretical computer science in which one tries to find the shortest route that passes through every point in a set, once and only once. The optimal solution to the problem for an arbitrary set of points cannot be found in a reasonable amount of time, that is, it cannot be found in polynomial time. However, we can find an approximation to the optimal solution in a reasonable amount of time. It would be best to find the approximate solution using a

- ☒ A. Heuristic algorithm
- ☐ B. Brute-Force algorithm
- ☐ C. Recursive algorithm
- ☐ D. Dynamic Programming

Time Taken 00:51

Prev Next

1 2 3 4 5 6 7 8 9 10

Finish Exam

Figure 9: An example timed exam. Notice that only one question is displayed at a time.

3.5 Practice Tool

YeckehZaare created and tested an adaptive practice tool on the Runestone platform [47] based on the theory of desirable difficulties [1]. Desirable difficulties are those that improve long-term retention, but reduce short-term learning gains. One example is spacing practice over many days rather than studying the night before an exam. Another is interleaving different topics rather than studying just one topic at a time. A third is forcing yourself to retrieve an answer to a question rather than just rereading the question and answer. The practice tool provides spaced, interleaved, and retrieval-based practice by reusing the practice questions in an ebook and presenting a question on a topic just before an algorithm (a modified version of SuperMemo 2) predicts that the learner is likely to forget that topic. One question is shown at a time as shown in Figure 10. Instructors can set up the practice tool to provide a number of points per completed practice session and can specify the number of practice sessions that learners must complete to earn the maximum number of points.

4 FEATURES FOR INSTRUCTORS

The Runestone platform allows instructors to create a custom course from an existing ebook, have students register for that custom course, create assignments from existing ebook materials, view student progress, grade assignments, and even author new problems for assignments. See [33] for the instructor's guide.

rec-5-44: What is the value of the following expression:

16 - 2 * 5 // 3 + 1

- ☒ A. 14
- ☐ B. 24
- ☐ C. 3
- ☐ D. 13.667

Check Me Compare me

Using parentheses, the expression is evaluated as (2*5) first, then (10 // 3), then (16-3), and then (13+1).

Done! Ask me another question!

I want to postpone this to tomorrow! Ask me another question.

Figure 10: The practice tool interface.

To become an instructor on the Runestone platform, first create a login on <https://runestone.academy>. Register for any of the existing ebooks. The URL for creating a custom course is in the instructor's guide. The form is shown in Figure 11. Enter a unique name in the project name field. This will be your custom course name.

Build a Custom Course

Logged in

This page allows you to select a book for your own class. Once you have selected and built the book, you will also be able to edit the table of contents to re-order or remove content, and create assignments for your students. These assignments will be published right in the book. As the instructor, you can grade your students' homework, and have access to a dashboard that allows you to monitor student progress. To begin, enter a project name below.

Project Name

UniqueName-Year

This will be the name your students use when they register. Make it one word, short (no spaces)

Description

Describe your course

Institution

Your school

Choose one of our ready-made books for your course:

- ☐ Foundations of Python Programming
- ☒ How to Think Like a Computer Scientist: Interactive Edition
- ☐ Problem Solving with Algorithms and Data Structures using Python, Brad Miller and David Ranum

Figure 11: Creating a custom course.

Once you have created a custom course you can have students register for that custom course. The students enter the unique course name. The link to this form is in the instructor's guide.

Click on the person icon at the top right of the screen in your custom course to get to the instructor's page. The instructor's page contains tabs to show student progress (as shown in Figure 12), administer the course, grade, create assignments, setup the practice tool, and view help/documentation.

Instructors can create assignments for students from the existing material in the ebook or even author new problems. The interface for creating assignments is shown in Figure 13. Instructors can add chapters or sections of chapters to an assignment. All of the activities in the selected sections are automatically added to the assignment as well. Instructors can control the number of points

Figure 12: The student progress page.

awarded, how the activities are graded, and which answer to grade (first, last, or best).

Chapter	Subchapter	Activity count	# required	Points	Auto-grade	Which to grade	Delete
Chapter 12 - A Computer can Make Decisions	Complex Conditionals (and, or, and not)	5	4	1	Interaction	Best answer	X
Chapter 12 - A Computer can Make Decisions	Using Multiple If statements	7	5	1	Interaction	Best answer	X

Figure 13: Creating a new assignment.

5 SUMMARY OF RESEARCH STUDIES

5.1 First Ebook Trial

Brad and his colleague David Ranum at Luther College created an interactive version of the *How to Think Like a Computer Scientist* ebook in 2011. Their ebook contained text, images, videos, executable and editable Python code, and a code visualizer (Philip Guo's Python Tutor) [23]. Brad and David trialed the interactive version of the this ebook in the fall of 2011 with 66 undergraduate students and gathered feedback through a survey. The survey provided evidence that the students were positive about the ebook. Sixty-one (90%) of the students said that they would want to use a similar ebook in another course [35].

5.2 Studies by CSLearning4U

The CSLearning4U group applied findings from educational psychology to interactive ebooks. In particular the group used worked examples [44] plus practice [46] and multiple-modalities (audio tours) [30]. The goal was to make learning more efficient to meet the needs of busy in-service secondary teachers. The group first added content to the *How to Think Like a Computer Scientist* ebook

and later created a teacher and student version of an ebook for the Advanced Placement (AP) Computer Science Principles (CSP) course. This course is offered in secondary schools and is intended to be equivalent to a college level course for non-majors. It covers programming fundamentals including variables, loops, conditionals, and functions.

The group conducted a usability survey with 18 teachers comparing several interactive features (code execution, code visualisation, Parsons problems, and multiple choice questions) on three ebook platforms (Runestone, Zyante, and CS Circles) [13]. The majority of the participants preferred the Runestone platform interface.

A log file analysis of usage of the *How to Think Like a Computer Scientist* ebook found that more users attempted Parsons problems than nearby multiple-choice questions, that the number of users declined from the start to the end of each chapter, and that users were more likely to attempt practice problems than watch videos, listen to audio tours, or edit code [16]. While most users were able to solve the Parsons problems in just one or two attempts, there were users who took between 20 and 100 attempts to solve each problem. Some users gave up and never solved the Parsons problems.

Ten teachers worked through the first eight chapters of the AP CSP ebook at their own pace [13]. Five of the ten teachers completed the first eight chapters for a 50% completion rate. This is higher than typical MOOC completion rates, however the teachers received a \$50 gift card for completing the chapters. The teachers who had the highest scores on the post-tests also used the interactive features the most [13].

A large-scale study with 130 teachers using version two of the teacher ebook found that teachers valued the worked example plus practice approach, Parsons problems, and having lots of practice problems [18]. Many teachers did not edit code or listen to audio tours. However, some teachers reported that they found audio tours useful for demonstrating how to describe code. Teachers reported that using the ebook increased their confidence in their ability to teach the content.

A log file analysis of the teacher and student versions of the AP CSP ebook provided evidence that teachers use the ebook differently than students [37]. Students ran the same program multiple times without changing it and made many more failed attempts to solve problems. Teachers may be more aware of successful and unsuccessful learning strategies.

5.3 Parsons Problems Research

Ericson used Runestone to study the efficiency and effectiveness of solving Parsons problems versus fixing code or writing code [17]. Undergraduate students solved Parsons problems significantly more quickly than fixing or writing the equivalent Python code and with comparable learning gains from pretest to post-test.

Ericson also tested the efficiency and effectiveness of solving adaptive Parsons problems, versus non-adaptive Parsons problems, versus writing the equivalent Python code [15]. Again, undergraduate students solved both types of Parsons problems significantly more quickly than writing the equivalent code. The adaptive Parsons problem group had significantly higher learning gains than the control group.

Ericson conducted an observational study with 11 secondary teachers solving both adaptive and non-adaptive Parsons problems [14]. All of the teachers were able to solve the adaptive problems, but some teachers could not solve some of the non-adaptive problems. Most of the teachers preferred adaptive Parsons problems. All of the teachers felt that solving Parsons problems helped them learn to fix and write similar code. Analysis of a log file with data from thousands of users provided evidence that learners are almost twice as likely to correctly solve adaptive Parsons problems than non-adaptive ones.

5.4 Practice Tool Research

Students tend to prefer techniques that improve short-term learning over those that improve long-term retention and recall [29]. However, a study with 193 undergraduate students found that every hour of using the practice tool was correlated with an increase in the average final exam grades of 1.04% [47]. Students were required to answer ten questions correctly to earn a point for that day and could earn points for up to 45 days of practice. If students were stuck on a practice question they could skip that question and it would be asked again another day. Students reported finding the tool useful and motivating. In fact, 62 (32%) of the students used the practice tool more than 45 days, even though they did not earn any additional points for doing so, which indicates that they found the tool valuable.

6 RECORDED DATA

Runestone records ebook interaction data and this data can be retrieved. Instructors can download a log file from the instructor's page. The log includes the date and time of the interaction (timestamp), the user id (sid), the directive (event), and additional information (act) based on the event type, and a unique id (div_id) for each problem in an ebook as shown in Figure 14.

timestamp	sid	event	act	div_id
8/21/17 23:54	27	Audio	Line-by-line Tour	lcf1
8/21/17 23:54	27	Audio	play	lcf1
8/21/17 23:54	27	Audio	closeWindow	lcf1
8/21/17 23:57	23	video	play	ants
8/21/17 23:57	25	mChoice	answer:3:correct	q2_2_1
8/21/17 23:57	25	mChoice	answer:1:correct	q2_2_2
8/21/17 23:58	25	mChoice	answer:3:no	q2_2_3
8/21/17 23:59	25	mChoice	answer:0:correct	q2_2_3
8/22/17 0:02	26	page	view	index.html
8/22/17 0:03	26	page	view	JavaBasics/
8/22/17 0:03	25	activecode	edit	lcf1
8/22/17 0:04	25	parsonsMove	start 0_1_0-5_0-6_0-2_3_0-4_0 c0	thirdClass
8/22/17 0:04	25	parsonsMove	move 5_0-6_0-2_3_0-4_0 0_1_0 c0	thirdClass
8/22/17 0:04	25	parsonsMove	move 5_0-6_0-4_0 0_1_0-2_3_0 c0	thirdClass
8/22/17 0:04	25	parsonsMove	move 5_0-6_0 0_1_0-2_3_0-4_0 c0	thirdClass
8/22/17 0:04	25	parsonsMove	move 6_0 0_1_0-2_3_0-4_0-5_0 c0	thirdClass
8/22/17 0:04	25	parsonsMove	move - 0_1_0-2_3_0-4_0-5_0-6_0 c0	thirdClass
8/22/17 0:04	25	parsons	correct - 0_1_0-2_3_0-4_0-5_0-6_0 c1-s	thirdClass

Figure 14: Part of a log file from Runestone that has been anonymized.

The log contains a record of each page view, code execution, code edit, video play, audio tour play, and practice question answer. All code that is executed or saved is available as well, but it does not appear in the standard log file. Interested instructors or researchers

can request the code log from Brad Miller, who runs the Runestone server. The log also records the answers to each of the practice questions as shown in Figure 14 for the multiple-choice questions (mchoice) and Parsons problems (parsons). The log also records every move of a block in a Parsons problem (parsonsMove).

7 FUTURE WORK

We plan to add support for code stepping and unit tests for Java, C, and C++. Unit tests provide immediate feedback on the correctness of code, which is important for learning [4]. Code stepping can improve student learning [24] and teaching [36].

The current instructor dashboard was designed for secondary teachers with small classes (30 max) and does not perform well for large college classes. We would like to improve the performance of the instructor dashboard and redesign it to better meet the needs of both college instructors and secondary teachers.

We would like to add a student dashboard to help students track their performance and compare their performance to others [2, 25, 45]. We would also like to add an author dashboard to help authors identify ways to improve their ebooks.

We plan to add support for peer instruction (PI) [7]. In PI, students answer a question individually and then discuss their answer with nearby students before answering again. PI has been shown to improve student understanding and engagement in many fields including physics [7], biology [26], and computer science [40, 41]. An effective PI question should be correctly answered by 40-60% of the students on their first attempt [27]. While there are some repositories of PI questions, they do not include this important data. We will analyze the log file data and record the percentage of learners who get each multiple-choice question correct on the first attempt. We will modify Runestone to allow instructors to search for suitable PI questions and add support for the PI process.

We encourage others to contribute to the Runestone platform by creating unit tests for existing problems, fixing problems, adding new features, or writing new interactive ebooks.

8 CONCLUSION

The Runestone ebook platform contains unique features that make it attractive for instructors, learners, authors and researchers. Instructors value the ability to create a custom course, manage their class, create assignments, view student progress, and grade assignments. Students value the interactivity, variety of practice types, immediate feedback, and price (free). Authors value the ease of publishing and revising ebooks, the platform's ability to easily handle 25,000+ students a day, and the growing author community. Researchers value having an environment that is extensible so that they can create new features and test them with thousands of learners.

ACKNOWLEDGMENTS

We thank all those who have contributed to the Runestone platform, authored ebooks, and the students and teachers who have used Runestone ebooks. Specifically, we thank Iman YekkehZaare for his work on the spaced practice tool. This work was supported by the National Science Foundation under grants 1138378 and 1432300.

REFERENCES

- [1] Elizabeth L Bjork, Robert A Bjork, et al. 2011. 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* 2, 59–68 (2011).
- [2] Robert Bodily and Katrien Verbert. 2017. Trends and issues in student-facing learning analytics reporting systems research. In *Proceedings of the seventh international learning analytics & knowledge conference*. ACM, 309–318.
- [3] Georg Brandl. 2010. Sphinx documentation. URL <http://sphinx-doc.org/sphinx.pdf> (2010).
- [4] John D Bransford, Ann L Brown, Rodney R Cocking, et al. 2000. *How people learn*. Vol. 11. Washington, DC: National academy press.
- [5] Peter Brusilovsky, Elmar Schwarz, and Gerhard Weber. 1997. Electronic textbooks on the world wide web: From static hypertext to interactivity and adaptivity. *Web based instruction* (1997), 255–261.
- [6] Nick Cheng and Brian Harrington. 2017. The Code Mangler: Evaluating Coding Ability Without Writing any Code. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 123–128.
- [7] Catherine H Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *American journal of physics* 69, 9 (2001), 970–977.
- [8] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*. ACM, 113–124.
- [9] Massimo Di Pierro. 2013. *web2py*. Lulu.com.
- [10] Alex Edgcomb and Frank Vahid. 2014. Effectiveness of online textbooks vs. interactive web-native content. In *2014 ASEE Annual Conference*.
- [11] Alex Edgcomb, Frank Vahid, Roman Lysecky, Andre Knoesen, Rajeevan Amirtharajah, and Mary Lou Dorf. 2015. Student performance improvement using interactive textbooks: A three-university cross-semester analysis. In *2015 ASEE Annual Conference and Exposition*.
- [12] Alex Edgcomb, Frank Vahid, Roman Lysecky, and Susan Lysecky. 2017. Getting students to earnestly do reading, studying, and homework in an introductory programming class. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 171–176.
- [13] Barbara Ericson, Mark Guzdial, Briana Morrison, Miranda Parker, Matthew Moldavan, and Lekha Surasani. 2015. An eBook for teachers learning CS principles. *ACM Inroads* 6, 4 (2015), 84–86.
- [14] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. ACM.
- [15] Barbara J. Ericson, James D. Foley, and Jochen Rick. 2018. Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research (ICER '18)*. ACM, New York, NY, USA, 60–68. <https://doi.org/10.1145/3230977.3231000>
- [16] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. 2015. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 169–178.
- [17] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling Conference on Computing Education Research*. ACM, 20–29.
- [18] Barbara J Ericson, Kantwon Rogers, Miranda Parker, Briana Morrison, and Mark Guzdial. 2016. Identifying design principles for CS teacher Ebooks through design-based research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, 191–200.
- [19] Eric Fouh, Ville Karavirta, Daniel A Breakiron, Sally Hamouda, Simin Hall, Thomas L Naps, and Clifford A Shaffer. 2014. Design and architecture of an interactive eTextbook—The OpenDSA system. *Science of Computer Programming* 88 (2014), 22–40.
- [20] David Goodger. 2019. reStructuredText. URL <http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html> (2019).
- [21] Scott Graham. 2011. Skulpt. URL <http://www.skulpt.org> (2011).
- [22] John Gruber. 2019. Markdown Guide. URL <https://www.markdownguide.org/basic-syntax/> (2019).
- [23] Philip J Guo. 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.
- [24] Christopher D Hundhausen, Sarah A Douglas, and John T Skasko. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing* 13, 3 (2002), 259–290.
- [25] Ioana Jivet, Maren Scheffel, Marcus Specht, and Hendrik Drachslers. 2018. License to evaluate: preparing learning analytics dashboards for educational practice. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. ACM, 31–40.
- [26] Jennifer K Knight and William B Wood. 2005. Teaching more by lecturing less. *Cell biology education* 4, 4 (2005), 298–310.
- [27] Nancy Kober. 2015. *Reaching students: What research says about effective instruction in undergraduate science and engineering*. National Academies Press.
- [28] Ari Korhonen, Thomas Naps, Charles Boisvert, Pilu Crescenzi, Ville Karavirta, Linda Mannila, Bradley Miller, Briana Morrison, Susan H Rodger, Rocky Ross, et al. 2013. 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*. ACM, 53–72.
- [29] Nate Kornell and Robert A Bjork. 2009. A stability bias in human memory: Overestimating remembering and underestimating learning. *Journal of experimental psychology: General* 138, 4 (2009), 449.
- [30] Richard E Mayer. 2008. Applying the science of learning: Evidence-based principles for the design of multimedia instruction. *American psychologist* 63, 8 (2008), 760.
- [31] Richard E Mayer and Roxana Moreno. 1998. A split-attention effect in multimedia learning: Evidence for dual processing systems in working memory. *Journal of educational psychology* 90, 2 (1998), 312.
- [32] Brad Miller. 2018. An Overview of Runestone Interactive. URL <https://runestone.academy/runestone/static/overview/overview.html> (2018).
- [33] Brad Miller. 2018. Using eBooks with Runestone Interactive. URL <https://runestone.academy/runestone/static/instructorguide/index.html> (2018).
- [34] Brad Miller and David Ranum. 2014. Runestone interactive: tools for creating interactive course materials. In *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 213–214.
- [35] Bradley N Miller and David L Ranum. 2012. Beyond PDF and ePub: toward an interactive textbook. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. ACM, 150–155.
- [36] Thomas L Naps, Guido Rößling, Vicki Almström, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. 2002. Exploring the role of visualization and engagement in computer science education. In *ACM Sigcse Bulletin*, Vol. 35. ACM, 131–152.
- [37] Miranda C Parker, Kantwon Rogers, Barbara J Ericson, and Mark Guzdial. 2017. Students and Teachers Use An Online AP CS Principles eBook Differently: Teacher Behavior Consistent with Expert Learners. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, 101–109.
- [38] Dale Parsons and Patricia Haden. 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—Volume 52*. Australian Computer Society, Inc., 157–163.
- [39] Kerttu Pollari-Malmi, Julio Guerra, Peter Brusilovsky, Lauri Malmi, and Teemu Sirkä. 2017. On the value of using an interactive electronic textbook in an introductory programming course. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. ACM, 168–172.
- [40] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer instruction: do students really learn from peer discussion in computing? In *Proceedings of the seventh international workshop on Computing education research*. ACM, 45–52.
- [41] Leo Porter, Dennis Bouvier, Quintin Cutts, Scott Grissom, Cynthia Lee, Robert McCartney, Daniel Zingaro, and Beth Simon. 2016. A multi-institutional study of peer instruction in introductory computing. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 358–363.
- [42] Clifford A Shaffer. 2016. Opensda: An interactive etextbook for computer science courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 5–5.
- [43] Clifford A Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L Naps. 2011. Opensda: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. ACM, 112–117.
- [44] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.
- [45] Stephanie Danell Teasley. 2019. Learning analytics: where information science and the learning sciences meet. *Information and Learning Sciences* 120, 1/2 (2019), 59–73.
- [46] John Gregory Trafton and Brian J Reiser. 1994. *The contributions of studying examples and solving problems to skill acquisition*. Ph.D. Dissertation. Citeseer.
- [47] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. 2019. A Spaced, Interleaved Retrieval Practice Tool that is Motivating and Effective. In *Proceedings of the International Computing Education Research Conference (ICER '19)*, August 12–14, 2019, Toronto, ON, Canada. ACM.
- [48] Yin Zhang and Sonali Kudva. 2013. Ebooks vs. print books: Readers’ choices and preferences across contexts. In *Proceedings of the 76th ASIS&T Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries*. American Society for Information Science, 108.