Towards Modeling Student Engagement with Interactive Computing Textbooks: An Empirical Study

David H. Smith IV
University of Illinois Urbana-Champaign
dhsmith2@illinois.edu

Qiang Hao
Western Washington University
qiang.hao@wwu.edu

Christopher D. Hundhausen
Washington State University
hundhaus@wsu.edu

Filip Jagodzinski
Western Washington University
jagodzf@wwu.edu

Josh Myers-Dean
Western Washington University
myersdj@wwu.edu

Kira Jaeger
Western Washington University
jaegerk2@wwu.edu

ABSTRACT

Interactive textbooks have great potential to increase student engagement with the course content which is critical to effective learning in computing education. Prior research on digital textbooks and interactive visualizations contributes to our understanding of student interactions with visualizations and modeling textbook knowledge concepts. However, research investigating student usage of interactive computing textbooks is still lacking. This study seeks to fill this gap by modeling student engagement with a Jupyter-notebook-based interactive textbook. Our findings suggest that students' active interactions with the presented interactive textbook, including changing, adding, and executing code in addition to manipulating visualizations, are significantly stronger in predicting student performance than conventional reading metrics. Our findings contribute to a deeper understanding of student interactions with interactive textbooks and provide guidance on the effective usage of said textbooks in computing education.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**;

KEYWORDS

Jupyter notebook, interactive textbook, visualization, learning analytics, interaction behavior modeling

ACM Reference Format:
David H. Smith IV, Qiang Hao, Christopher D. Hundhausen, Filip Jagodzinski, Josh Myers-Dean, and Kira Jaeger. 2021. Towards Modeling Student Engagement with Interactive Computing Textbooks: An Empirical Study. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432361

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 '21, March 13–20, 2021, Virtual Event, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432361

1 INTRODUCTION

Computing educators have long shared the vision of an interactive textbook platform that seamlessly integrates images, executable code, visualizations, and assessment components. This can be attributed to the field's emphasis on hands-on practice [1–4] and requirement for students to constantly solve problems through programming. These tasks require students to bridge the intellectual gap between the abstract concepts they learn in class and their concrete implementations, [5, 6], a task with which novices in particular have difficulty [7, 8]. Presenting both abstract and concrete information in a tightly integrated, interactive form is done with the intent of allowing students to learn these relationships through experimentation with interactive elements.

Though textbooks have long existed as a reliable learning resource it was not until recently that technologies mature enough to support interactive textbooks emerged [9]. Tools such as computational notebooks make the integration of static and dynamic elements seamless and streamline the process of creating such documents. A computational notebook is a document that can be read like a regular book chapter and executed like a computer program [10, 11]. The executable code can be used to generate a variety of other elements, such as mathematical formulas, dynamic visualizations, multiple-choice questions, and code writing exercises. Although computational notebooks themselves are far from new, their adoption as platforms for educational content delivery is a recent occurrence [12].

Much of the prior work on digital textbooks and interactive visualizations are relevant to investigations on interactive computing textbooks. Some notable points of focus include comparing the mediums of textbooks (e.g digital vs paper), modeling textbook knowledge concepts, modeling student reading patterns, and investigating student engagement with interactive visualizations [5, 6, 13–16]. These efforts contribute to our understanding of how to build adaptive learning modules, promote personalized learning, and examine techniques that model student reading. The research on student engagement with visualizations laid the foundation for investigating student engagement with interactive learning elements in the context of computing education [1].

Despite the contributions of prior studies, research that investigates student usage of interactive computing textbooks is still lacking. Prior studies on digital textbooks were typically agnostic to discipline when modeling student reading behaviors which did not account for the hands-on nature of computing education

[15, 17]. Previous investigations into interactive textbooks failed to engage in a fine-grained analysis of student interactions and their relationship with student learning outcomes [18, 19].

This study seeks to fill this gap by modeling student engagement with an interactive textbook that allows students to fully engage with and modify all interactive and static components. The results of this study contribute to a deeper understanding of (1) how students engage with interactive textbooks through reading and rich interactions, and (2) how students perceive the functionalities of interactive computing textbooks. Our findings shed light on how students engage with a fully interactive textbook, as well as inform computing educators of the effective usage of interactive computing textbooks.

2 BACKGROUND

2.1 Early Digital Textbooks

Early studies on digital textbooks focused on comparing the impacts of different mediums of textbooks (digital vs paper) on student learning [20, 21]. Findings showed that, regardless of medium, reading textbooks have been found to have positive impacts on student academic success [13, 20, 22–24]. Modeling reading patterns makes it possible to understand student reading behavior, which contributes to identifying gaps in student knowledge and recommending appropriate learning resources.

Research on modeling knowledge components in textbooks and student reading behaviors mainly focused on data that are relevant to passive reading, such as reading speed, jumping to the next page, and annotations [17]. However, reading speed as a metric for modeling reading has been found to have a weak relationship with student performance suggesting it may not be an accurate measure of learning [15]. Additionally, digital textbooks are becoming increasingly interactive, allowing a variety of heterogeneous activities such as quizzes, dynamic visualizations, and even problem-solving activities. These interactive activities are becomming an integral componenet of digital textbooks in many disciplines, such as computing education.

2.2 Interactive Visualization in Computing Education

Computer-based dynamic visualization as a medium for teaching has enjoyed a long history in computing and general STEM education with the earliest examples dating back to the 1980s [25, 26]. Interactive visualizations still enjoy wide popularity in computing education today as a method of providing a concrete, visual representation of otherwise abstract concepts [6]. They have also been shown to have positive effects on students in terms of performance, motivation, and engagement [27–29].

Researchers have sought to understand how students interact with visualizations systematically. Naps *et al.* [30] adapted the Bloom's Taxonomy [31] and defined an engagement taxonomy (ET) consisting of 5 levels: (1) No viewing - visualization is not used, (2) Viewing - visualization is watched, (3) Responding - questions related to visualization are answered, (4) Changing - different input data is provided by the learner to change visualization from its original state, (5) Constructing - students build a visualization, and (6) Presenting - students write or talk about visualizations.

Rößling's engagement taxonomy, Sorva's "Two Dimensional Engagement Taxonomy" and Myller's "Extended Engagement Taxonomy" have been notable extensions to Naps's each of which introduced more granularity and more focus on the direct student-visualization interactions that take place [29, 32, 33]. Myller's noteably distinguished between *direct engagement* and *content ownership*. The former is concerned with the level of interaction between the user and the visualization, with the latter focusing on the relationship between the student and the code that produces the visualization.

As interaction technology advances, researchers tend to differentiate student interactions with visualizations at different levels. This relates to the idea that the acts of modification and creation indicate a higher order conceptual understanding of the curriculum. The implication being that, the greater the agency of students to manifest their ideas through a given learning material, the greater their ability will be to leverage that material for learning.

2.3 Students and Interactive Textbooks

Table 1: Interactive Textbook Platforms and their Features

	Jupyter	OpenDSA	zyBooks	Runestone
Fully Modifiable	yes	no	no	no
Notes	yes	no	no	no
Visualizations	yes	yes	yes	yes
Slides	no	yes	no	no
Animations	yes	yes	yes	yes
Practice Sets	yes	yes	yes	yes
Code Sandbox	yes	no	yes	yes
Videos	yes	no	yes	yes
Quizzes	yes	yes	yes	yes
Open Access	yes	yes	no	yes

There have been multiple successful attempts to create an interactive textbook. These include popular web-native, open-source options such as Virginia Tech's OpenDSA project, and Runestone Interactive, in addition to paid solutions such as Wiley's zyBooks.

Studies of these platforms show that students overwhelmingly prefer them to traditional textbooks [34, 35]. Additionally, students are more likely to engage with visualizations and interactive elements over static text [36] and that these elements lead to increased motivation [37].

Interactive textbooks have also been shown to have a substantial impact on student performance. In a multi-institutional study, Edgecomb *et al.* noted a significant performance increase in classes that switched to zyBooks, particularly within the classes' lower quartile [18]. McKinny *et al.* corroborated these findings in addition to finding a significant increase in the pass rate compared to previous terms (78% -> 91%) [19].

Though there are instances of Jupyter Notebook being used to deliver course content in AI education [12] there are no studies, insofar as we are aware, of notebook usage in introductory computing education. Though existing interactive textbooks provide a significant number of features, they lack the extensibility of Jupyter

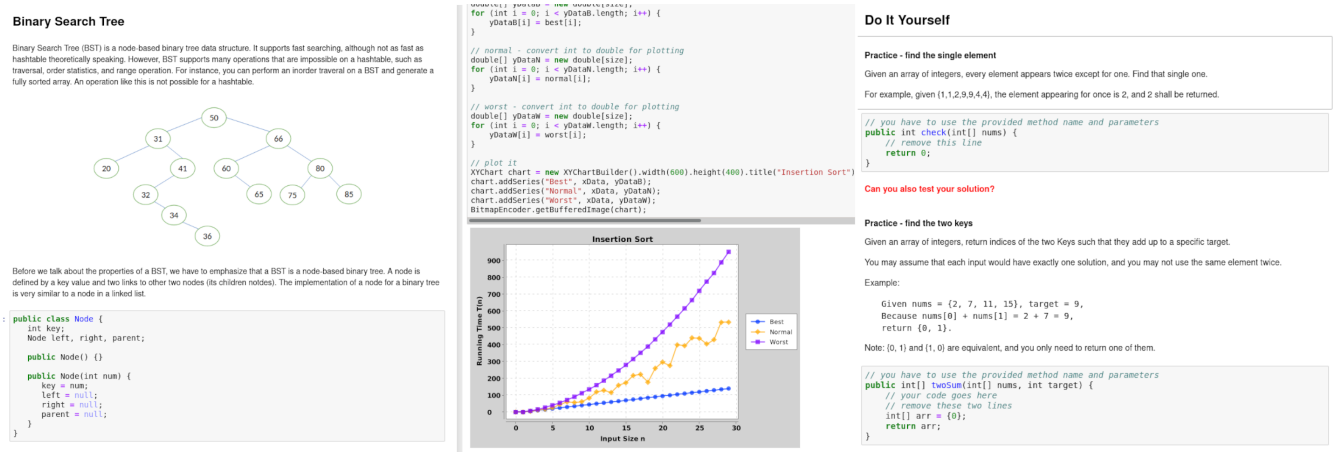

Figure 1: Various interactive components in one book chapter from the used textbook.

Notebook and limit how students can modify content and experiment within them (Table 1).

Overall, despite the contributions of prior studies, research that investigate student usage of interactive computing textbooks is still lacking. Work on digital textbooks were typically agnostic to disciplines in terms of modeling student reading behaviors, which did not account for the hands-on nature of computing education [15, 17]. Prior studies on students' interactions with interactive textbooks did not investigate, on a fine grained level, students' interactions with integrated static and visual components [18, 19]. To fill the gaps, we seek to investigate (1) CS student interaction with various components of a tested interactive textbook, (2) the relationship between such interactions and student academic performance, as well as (3) student learning experience of using such an interactive textbook.

3 RESEARCH DESIGN

3.1 Research questions

The research questions that guided this study include:

- **RQ1**: To what extent do students engage with the interactive textbooks?
- **RQ2**: To what extent does student engagement with interactive computing textbooks predict student performance?
- **RQ3**: What are student attitudes towards using interactive computing textbooks for learning?

3.2 Experimental Design

We conducted an in-class experiment in an introductory programming course that enrolled eighty students at a large university in the North American Pacific Northwest. Jupyter-notebook-based textbooks tailored to the course were deployed to JupyterHub to support online access and Travis-CI was used to support unit testing for the embedded practice problems. The textbook was organized by chapters of topics (e.g. stack, and queue, binary trees) and each chapter is represented as a Jupyter-notebook file, which interweaves static tests, images, and interactive code along with dynamic visualizations and images that are generated from executable code [12]. Additionally, each topic is supported by a set of relevant practice coding questions (see Figure 1).

Students were required to read a specific chapter from a Jupyter-notebook-based textbook before and after each lecture. During the lecture, the same knowledge on a more fine-grained level was delivered, discussed, and practiced. Additionally, students were encouraged to use the textbook for quick code testing and note-taking during lectures.

3.3 Measurements and Data Collection

In answering **RQ1** and **RQ2** we must first define our measurements for student performance and student engagement. Students' conceptual understanding was measured through two, paper-based exams with their ability to problem-solve and implement solutions to being measured by programming assignments. Six complex programming assignments were given throughout the course to measure students' implementation abilities.

To measure student engagement we employed the use of Rößling's [33] adaptation of Naps' taxonomy [30]. This includes three levels of engagement: (1) Viewing - Reading Through the provided static text and static graphics, (2) Responding - responding to informal assessments (e.g. quizzes, coding problems), and (3) Changing - manipulating the dynamic visualization by updating the code (e.g. changing algorithm parameters) or creating a new visualization. Guided by this engagement taxonomy, we collected the following three data sets corresponding to the three levels of engagement:

- **Reading time:** The amount of time a student has a notebook open in their browser and can spend reading or referencing it.
- **Response Frequency:** The number of times students executed cells containing the provided code, quizzes, and coding problems per chapter.
- **Change Frequency:** The number of times students manipulated the provided visualization and creating new code cells for different purposes beyond responding to the given questions per chapter.

Demographic and course history data was collected through surveys to account for the influence of individual differences in outcomes.

To address **RQ3**, we surveyed all participating students using three open-ended questions:

- How did you use the Jupyter-notebook-based textbook?
- What do you like about the Jupyter-notebook-based textbook?
- What do you dislike about the Jupyter-notebook-based textbook?

4 RESULTS

4.1 RQ1: To what extent do students engage with the interactive textbooks?

Tracking showed that students, on average, spent 4.12 hours with the provided textbook open per week (*reading time*), responding to embedded quizzes or code writing questions for 6.16 times per week (*response frequency*), and changing and executing visualization code or write unrequired code for 5.94 times per week (*change frequency*) (see Figure 2).

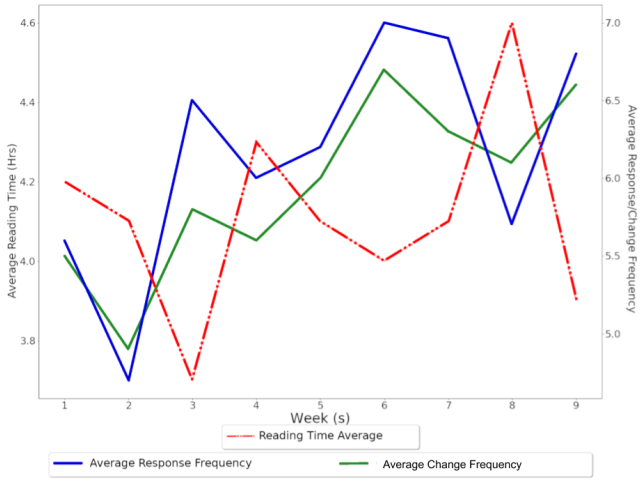

Figure 2: The trend of tracked reading time, response frequency, and change frequency.

As is shown in Figure 2, students tended to spend more time on reading the textbook and interacting with the textbook more frequently when the course progressed towards more complicated topics. The two types of tracked interactions, *response frequency* and *change frequency*, seem to have a consistent pattern over time. Although the tracked reading time seemed to follow the general trend, there was a notable difference between its pattern and the patterns of the two types of interactions.

To further understand the relationships between *reading time* and other interactions we performed Pearson's correlation analysis on the three tracked data. *response frequency* and *change frequency* showed a high degree of correlation (r=0.68, p<0.01) with *reading time* being correlated with *change frequency* to a limited extent (Table 2). Though this finding demonstrates some consistency between reading time and interactions, those differences that do exist may indicate reading time alone does not give a full representation of student engagement with the textbooks.

Table 2: The correlation relations among reading time, frequency of responses, and frequency of changes.

	Reading Time (RT)	Response Frequency (RF)	Change Frequency (CF)
RT	1.00	-	-
RF	0.25	1.00	-
CF	0.36*	0.68**	1.00

*p < 0.05; **p < 0.01; ***p < 0.001.

4.2 RQ2: To what extent does student engagement with interactive textbooks predict student performance?

To understand the predictive power of student engagement with interactive textbooks on their performance, we conducted blockwise regression on two sets of factors. The first set of factors were control factors, including the demographics of students (gender and race) and the number of CS courses students have taken in high school and college before taking the current course. Given the lack of standardization that exists within high-school CS curriculum (and possibly community college), we used only the counts of courses in our analysis. The control factors explained 6.1% of the variance of student performance (R^2 = 0.06, p > 0.01). The predictive power of control factors is presented in Table 3.

Table 3: Multiple regression analysis on student performance using control factors.

	R^2	R^2 adj.	F	β	t
Student Performance	0.06	0.01	1.29		
Control Factors:					
Gender				-0.54	-1.76
Race				0.08	0.60
CS courses before college				-0.10	-0.70
CS courses during college				0.04	0.30

*p < 0.05; **p < 0.01; ***p < 0.001.

The second set of factors was the focus of this study, including *reading time*, *response frequency*, and *change frequency*. When *reading time* was added as the only factor in the focus block to the regression model, the variance explained by all factors for student performance increased to 10.0% (R^2 = 0.10, p > 0.05). There was no significant difference (F=2.46, p > 0.05) between this new model (see Table 4) and the one reported in Table 3.

No or little multicollinearity is an important assumption of regression analysis. Given the strong correlations between *response frequency* and *change frequency*, we could not directly add both of them to the model. As a result, we performed Principal Component Analysis (PCA) to extract a single factor from the two. The Kaiser-Meyer-Olkin (KMO) measure and Bartlett's test of sphericity were applied to verify the validity of PCA. The sampling adequacy was verified for both factors, with all KMO values bigger than 0.5. The

Table 4: Multiple regression analysis on student performance using control factors and reading time.

	R^2	R^2 adj.	F	β	t
Student performance	0.10	0.03	1.55		
Control factors:					
Gender				-0.49	-1.61
Race				0.08	0.68
CS courses before college				-0.08	-0.61
CS courses during college				0.05	0.34
Focus:					
Reading time				0.19	1.57

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

results of Bartlett's test of sphericity were significant, indicating that correlations between items were sufficiently large for both factors. We named the extracted single factor *interaction*.

When the *interaction* factor was added as the second factor of the focus block to the model, the variance explained by all factors for student performance increased to 42.0% (R^2 = 0.42, $p < 0.05$). Analysis of variance was used to compare this model (see Table 5) and the one reported in Table 4, and found a significant difference (F=40.71, $p < 0.01$) between the two models.

Table 5: Multiple regression analysis on student performance using control factors and two focus factors.

	R^2	R^2 adj.	F	β	t
Student performance	0.42	0.37	8.77		
Control factors:					
Gender				-0.46	-1.90
Race				0.03	0.29
CS courses before college				-0.08	-0.76
CS courses during college				-0.04	-0.35
Focus:					
Reading time				0.03	0.38
Interaction				0.54**	6.38

$^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$.

4.3 RQ3: What are student attitudes towards using interactive textbooks for learning?

To understand student attitudes towards using an interactive textbook, two raters (a trained undergraduate student and an experienced computing education researcher) iteratively categorized the answers to each survey question into major themes using an inductive analysis approach [38]. The average inter-rater reliability (Cohen's Kappa) across the three questions was 0.84. The finalized coding scheme and individual question's coding reliability are presented in Table 6. For the differences between the categorization of the same response to a question, the two raters would further discuss with each other and make a final decision collectively.

In answering the first question, **"How did you use the Jupyter-Notebook-based textbook?"**, all students expressed their usage was similar to how they would use a traditional digital textbook. They previewed and reviewed knowledge present in lectures in addition to attempting the embedded daily practices. It is worth noting that many students mentioned that they achieve such goals by actively interacting with the textbook, such as "*stepping through the provided code*", "*playing with the actual code*", and "*trying out my ideas through code*".

Table 6: The finalized coding scheme for analyzing student responses to our survey questions.

Questions	Categories	Code	k
Q1	Previewing	1. read before class 2. code before class 3. practice before class	0.79
	Revieweing	1. read during lectures 2. notes during lectures 3. code during lectures	
	Practicing	1. practice before class 2. practice after class	
Q2	Integration	1. execute code 2. visualization 3. write code 4. manipulate visualization	0.88
Q3	Functionality	1. navigation 2. code execution 3. manipulate visualization 4. error messages	0.85
	Efficiency	1. navigation difficulty 2. code execution difficulty 3. visualization interaction difficulty 4. debugging difficulty	

The second question was **"What do you like about the Jupyter-Notebook-based textbook?"**. Students uniformly expressed their enjoyment of the seamless combination of code, text, and visualization provided by computation notebooks. Student A said *"I love how versatile the textbook is. You can read the explanation, and try out the corresponding Java code, see the results without even compiling the code."*. Student B responded *"I like how straightforward it makes the time complexity for me to understand. It is just like a secondary less with actual code."*.

From the third question, **"What do you dislike about the Jupyter-Notebook-based text-book?"**, we found two major themes. The first regarding the functionality of the notebooks and the second relating to the efficiency of its configuration. Student C stated that "*I have to run all the (code) cells manually so I can run the cell that I really wish to. It's so tedious*". This was said in spite of the fact that, though cells are interdependent, they need not be executed one at a time. Jupyter Notebooks allow for the sequential execution of multiple cells through multiple selection or shortcuts accessible through the toolbar. This statement indicates that students may have had a limited understanding of the notebook and its available functionality. Student D commented that *"The limited error message makes coding in Jupyter Notebook difficult."*. Though Jupyter Notebooks provide some error messages, they lack the level of detail that is commonplace in most modern IDEs. This may provide an additional barrier to students learning the already difficult task of debugging.

5 DISCUSSION

5.1 Student Engagement

We highlight two findings of our study in terms of RQ1 and RQ2:

(1) The weak correlation between reading time and interactions suggests a difference in interaction patterns.
(2) Student interactions with the textbook have stronger predictive power than reading. This is evidenced by the comparison between the interaction and reading time factors in predicting student performance.

These findings are each in alignment with prior studies of student interactions with visualization. A meta-study by Hundhausen *et al.* [27] found that how students use visualization, rather than what they see, has the greatest impact on education effectiveness. The findings of this study, in the context of interactive textbooks, support this emphasis on student active engagement. After all, interactive technology is effective only when it is used to actively engage students [1, 28, 30, 39, 40]. However, it is worth noting that the capability of Jupyter-notebook-based textbooks substantially goes beyond algorithm visualization as well as that of most interactive textbooks (See 1). In the environment of Jupyter-notebook-based textbooks, students can manipulate the provided code that generates visualizations, write their own code for exploration or problem-solving, and see immediate results. These two findings provide supportive evidence for extending the "*Taxonomy of Student Engagement with Visualizations*" [30, 33] to account for the various types of engagement in the environment of digital notebooks.

The findings of this study also shed light on the use of reading time as a metric of engagement. Reading time is a conventional measurement of engagement that is associated with student learning and has had an influence on data tracking for digital textbooks [41, 42]. However, given the inconsistency between active interactions and reading time combined with the weak predictive power of reading time on student performance, reading time may not be the best measure of student engagement. Analysis of plots showing interactions over time (See Figure 2) and blockwise regression suggest that student interactions with the textbook are a greater predictor of student performance compared to reading time.

5.2 Students' perception of interactive textbooks

Jupyter-notebook-based textbooks provide an environment that integrates text, example code, visualizations, and interactive widgets. This integrated environment empowers students to (1) step through text and example code, (2) manipulate visualization, and (3) test their ideas through executable code. This integrated environment provides students with a degree of engagement that paper-based textbooks can not afford and was universally preferred by the majority of participants in this study.

Despite these advantages, it is worth noting that Jupyter Notebook was originally designed for professionals to conduct data analysis exploration and document and share the analysis process, rather than developing and hosting digital textbooks [43]. As a result, some features that are acceptable to professionals may be difficult for novice learners in computer science. Many participating students pointed out the limited functionality of debugging in Jupyter Notebooks as a challenge for them to smoothly use the provided textbook. The limited debugging functionality, in combination with the lack of line numbers in code cells, adds unnecessary difficulty to the already arduous task of debugging. Computational notebooks have great potential to be an excellent tool for creating interactive textbooks in fields such as computer and data science. The inclusion of a more powerful suite of debugging tools may better fulfill this purpose.

Another challenge faced by the participating students was their lack of familiarity with Jupyter Notebooks despite ten weeks of continuous usage. Some students indicated that they had to manually execute each code cell sequentially and expressed frustration over the seemingly tedious process. This process can actually be fully automated by clicking a button on the navigation bar of a notebook page. This finding indicates that instructors may not assume that computational notebooks are sufficiently intuitive to students. Instead, instructors should consider providing an in-depth overview of the computational notebooks at the beginning of the course so that students can use Jupyter-notebook-based textbooks to their full capabilities.

6 LIMITATIONS AND THREATS TO VALIDITY

The first and most critical limitation of this study is our operationalization of reading time. This study tracked the active time of a browser tab with an open book chapter and assumed that the tracked time is student reading time. Although this is a common practice in tracking student reading activity there is evidence it may not be an accurate reflection of student reading time [15]. Future studies may consider exploring new approaches to reading time tracking, such as using periodical prompts that ask users to indicate ongoing reading activities.

Additionally, this study was implemented in a course that was particularly well suited for interactive textbooks and visualizations. The class employed a flipped teaching approach which gives students particularly strong incentives for actively engaging with the textbooks. Classes that employ different pedagogical practices or are not tightly integrated with textbook material may experience different results. Future studies may consider replicating this study on a larger scale and investigating the factors that contribute to student engagement with interactive textbooks and how these are related to the environment in which they are presented [44].

7 CONCLUSIONS

This study explored modeling student engagement with an interactive digital computing textbook. We found that the pattern of active interaction did not align with that of reading time. We also found that active interaction was significantly more effective than reading time in predicting student performance. Additionally, students' feedback on using interactive textbooks shed light on the effective practice of adopting computational notebooks in educational contexts and showed potential functionality improvements for using computational notebooks as interactive textbook platforms. The results of this study contribute to a deeper understanding of student interactions with computational notebook based digital textbooks and provide guidance on the effective usage of such textbooks.

REFERENCES

[1] Eric Fouh, Monika Akbar, and Clifford A Shaffer. The role of visualization in computer science education. *Computers in the Schools*, 29(1-2):95–117, 2012.

[2] Shirley Booth. Learning computer science and engineering in context. *Computer Science Education*, 11(3):169–188, 2001.

[3] Qiang Hao, Brad Barnes, Robert Maribe Branch, and Ewan Wright. Predicting computer science students' online help-seeking tendencies. *Knowledge Management & E-Learning: An International Journal*, 9(1):19–32, 2017.

[4] David H Smith IV, Qiang Hao, Vanessa Dennen, Michail Tsikerdekis, Bradly Barnes, Lilu Martin, and Nathan Tresham. Towards understanding online question & answer interactions and their effects on student performance in large-scale stem classes. *International Journal of Educational Technology in Higher Education*, 17(1):1–15, 2020.

[5] Andreas Holzinger, Michael Kickmeier-Rust, and Dietrich Albert. Dynamic media in computer science education; content complexity and learning performance: is less more? *Journal of Educational Technology & Society*, 11(1), 2008.

[6] Clifford A Shaffer, Matthew L Cooper, Alexander Joel D Alon, Monika Akbar, Michael Stewart, Sean Ponce, and Stephen H Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE)*, 10(3):9, 2010.

[7] Anna Eckerdal and Michael Thuné. Novice java programmers' conceptions of" object" and" class", and variation theory. *ACM SIGCSE Bulletin*, 37(3):89–93, 2005.

[8] David H Smith IV, Qiang Hao, Filip Jagodzinski, Yan Liu, and Vishal Gupta. Quantifying the effects of prior knowledge in entry-level programming courses. In *Proceedings of the ACM Conference on Global Computing Education*, pages 30–36. ACM, 2019.

[9] Diane Pecorari, Philip Shaw, Aileen Irvine, Hans Malmström, and Špela Mežek. Reading in tertiary education: Undergraduate student practices and attitudes. *Quality in Higher Education*, 18(2):235–256, 2012.

[10] Fernando Perez and Brian E Granger. Project jupyter: Computational narratives as the engine of collaborative data science. *Retrieved September*, 11(207):108, 2015.

[11] Adam Rule, Aurélien Tabard, and James D Hollan. Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 32. ACM, 2018.

[12] Keith O'Hara, Douglas Blank, and James Marshall. Computational notebooks for ai education. In *The Twenty-Eighth International Flairs Conference*, 2015.

[13] David B Daniel and William Douglas Woody. E-textbooks at what cost? performance and use of electronic v. print texts. *Computers & Education*, 62:18–23, 2013.

[14] Ivica Boticki, Gökhan Akçapınar, and Hiroaki Ogata. E-book user modelling through learning analytics: the case of learner engagement and reading styles. *Interactive Learning Environments*, 27(5-6):754–765, 2019.

[15] Khushboo Thaker, Yun Huang, Peter Brusilovsky, and He Daqing. Dynamic knowledge modeling with heterogeneous activities for adaptive textbooks. In *The 11th International Conference on Educational Data Mining*, pages 592–595, 2018.

[16] Hemalatha Sasidharakurup, Rakhi Radhamani, Dhanush Kumar, Nijin Nizar, Krishnashree Achuthan, and Shyam Diwakar. Using virtual laboratories as interactive textbooks: Studies on blended learning in biotechnology classrooms. *ICST Trans. e-Education e-Learning*, 2(6):e4, 2015.

[17] Hiroaki Ogata, Chengjiu Yin, Misato Oi, Fumiya Okubo, Atsushi Shimada, Kentaro Kojima, and Masanori Yamada. E-book-based learning analytics in university education. In *International Conference on Computer in Education (ICCE 2015)*, pages 401–406, 2015.

[18] Alex Edgcomb, Frank Vahid, Roman Lysecky, Andre Knoesen, Rajeevan Amirtharajah, and Mary Lou Dorf. Student performance improvement using interactive textbooks: A three-university cross-semester analysis. In *2015 ASEE Annual Conference and Exposition*, 2015.

[19] Dawn McKinney, Alex Daniel Edgcomb, Roman Lysecky, and Frank Vahid. Improving pass rates by switching from a passive to an active learning textbook in cs0. In *2020 ASEE Virtual Annual Conference Experience*, 2020.

[20] Carrie Spencer. Research on learners' preferences for reading from a printed text or from a computer screen. *Journal of Distance Education*, 21(1):33–50, 2006.

[21] Yoram Eshet-Alkalai and Nitza Geri. Does the medium affect the message? the influence of text representation format on critical thinking. *Human Systems Management*, 26(4):269–279, 2007.

[22] Amanda J Rockinson-Szapkiw, Jennifer Courduff, Kimberly Carter, and David Bennett. Electronic versus traditional print textbooks: A comparison study on the influence of university students' learning. *Computers & Education*, 63:259–266, 2013.

[23] Reynol Junco and Candrianna Clem. Predicting course outcomes with digital textbook usage data. *The Internet and Higher Education*, 27:54–63, 2015.

[24] R Eric Landrum, Regan AR Gurung, and Nathan Spann. Assessments of textbook usage and the relationship to student course performance. *College Teaching*, 60 (1):17–24, 2012.

[25] Ronald Baecker. Sorting out sorting: A case study of software visualization for teaching computer science. *Software visualization: Programming as a multimedia experience*, 1:369–381, 1998.

[26] Marc H Brown and Robert Sedgewick. Techniques for algorithm animation. *Ieee Software*, (1):28–39, 1985.

[27] Christopher D Hundhausen, Sarah A Douglas, and John T Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[28] Christopher D Hundhausen and Jonathan L Brown. Designing, visualizing, and discussing algorithms within a cs 1 studio experience: An empirical study. *Computers & Education*, 50(1):301–326, 2008.

[29] Juha Sorva, Ville Karavirta, and Lauri Malmi. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4):1–64, 2013.

[30] Thomas Naps, Stephen Cooper, Boris Koldehofe, Charles Leska, Guido Rößling, Wanda Dann, Ari Korhonen, Lauri Malmi, Jarmo Rantakokko, Rockford J Ross, et al. Evaluating the educational impact of visualization. In *Acm sigcse bulletin*, volume 35, pages 124–136. ACM, 2003.

[31] Benjamin Samuel Bloom. *Taxonomy of educational objectives: the classification of educational goals: Handbook I, Cognitive domain*. McKay, 1969.

[32] Niko Myller, Roman Bednarik, Erkki Sutinen, and Mordechai Ben-Ari. Extending the engagement taxonomy: Software visualization and collaborative learning. *ACM Transactions on Computing Education (TOCE)*, 9(1):1–27, 2009.

[33] Guido Rößling, Thomas Naps, Mark S Hall, Ville Karavirta, Andreas Kerren, Charles Leska, Andrés Moreno, Rainer Oechsle, Susan H Rodger, Jaime Urquiza-Fuentes, et al. Merging interactive visualizations with hypertextbooks and course management. In *ACM SIGCSE Bulletin*, volume 38, pages 166–181. ACM, 2006.

[34] Tommy Färnqvist, Fredrik Heintz, Patrick Lambrix, Linda Mannila, and Chunyan Wang. Supporting active learning by introducing an interactive teaching tool in a data structures and algorithms course. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 663–668, 2016.

[35] Bradley N Miller and David L Ranum. Beyond pdf and epub: toward an interactive textbook. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pages 150–155, 2012.

[36] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 169–178, 2015.

[37] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. A spaced, interleaved retrieval practice tool that is motivating and effective. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, pages 71–79, 2019.

[38] Juliet M. Corbin and Anselm L. Strauss. *Basics of qualitative research: techniques and procedures for developing grounded theory*. SAGE Publications, Inc., 2008.

[39] David Furcy. Jhavepop: Visualizing linked-list operations in c++ and java. *Journal of Computing Sciences in Colleges*, 25(1):32–41, 2009.

[40] S Sriadhi, Robbi Rahim, and Ansari Saleh Ahmar. Rc4 algorithm visualization for cryptography education. In *Journal of Physics: Conference Series*, volume 1028, page 012057. IOP Publishing, 2018.

[41] Naomi S Baron. *Words onscreen: The fate of reading in a digital world*. Oxford University Press, USA, 2015.

[42] Yueh-Min Huang and Tsung-Ho Liang. A technique for tracking the reading rate to identify the e-book reading behaviors and comprehension outcomes of elementary school students. *British Journal of Educational Technology*, 46(4): 864–876, 2015.

[43] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

[44] Qiang Hao, David H Smith IV, Naitra Iriumi, Michail Tsikerdekis, and Amy J Ko. A systematic investigation of replications in computing education research. *ACM Transactions on Computing Education (TOCE)*, 19(4):1–18, 2019.