On the Value of Using an Interactive Electronic Textbook in an Introductory Programming Course

Kerttu Pollari-Malmi
Aalto University
kerttu.pollari-malmi@aalto.fi

Julio Guerra
Universidad Austral de Chile
mallium@gmail.com

Peter Brusilovsky
University of Pittsburgh
peterb@pitt.edu

Lauri Malmi
Aalto University
lauri.malmi@aalto.fi

Teemu Sirkiä
Aalto University
teemu.sirkia@aalto.fi

ABSTRACT

E-books including interactive elements are rapidly becoming more popular and are likely to largely replace traditional textbooks at university level education. In this paper, we report our initial observations on the changes we noticed in students' motivational factors and learning results when a static PDF textbook was replaced by an interactive e-textbook in a large CS1 service course. We found increase in both motivational factors, as well as learning gain. In addition, students' feedback on the learning resources improved. While the changes were not large, they encourage to continue integrating more interactive learning content into course learning environment.

CCS CONCEPTS

• **Applied computing** → *E-learning*;

KEYWORDS

CS1, programming education, E-textbooks, motivation

1 INTRODUCTION

Online and blended forms of education are rapidly changing the educational culture world wide because of the easy accessibility of free or low cost learning resources through Internet. Massive open online courses (MOOCs) have become mainstream as one important area in complementary and higher education. Self-study resources available, for example, in Khan Academy[1], Lynda[2], Youtube or Wikipedia provide vast opportunites for students to learn different topics. Yet these sources are widely based on short texts, video materials, and possibly interactive demonstrations. Complete electronic textbooks, e-textbooks, which provide a comprehensive resource for some topic with different *integrated interactive components*, instead of plain static PDF resources, or cross-linked static resources available, e.g., in Wikipedia, are more rare. However, considerable development work is carried out in this area. Some notable examples in the area of computing education include the CS principles textbook[3] by Guzdial and Ericson and OpenDSA textbook[4] for data structures and algorithms, by Shaffer et. al. [20].

In the area of computer science education the work in e-textbooks has largely focused on developing the necessary platforms and software to enable the interactive components, as well as creating the actual learning content for some target topic. So far there is little research which has explored the *impact of these resources on students' studying and learning*. It is in this area, where our current paper contributes.

In the next sections, we present an overview of relevant work in e-textbooks, followed by the description of our large scale target course in programming, with over 600 students. In Section 4, we explain our study, in which we explored the impact of our distributed e-textbook on students' motivation and learning results. Our results are preliminary and we discuss the directions of future research in the final section.

2 PREVIOUS RESEARCH

The research on electronic textbooks always concentrated on expanding the borders of traditional textbooks, focusing on various aspects that the traditional form was not able to support. The early work on electronic textbooks was associated with the development of hypertext with a pioneer electronic book called SuperBook [17] being one of the early hypertext projects. The use of hypertext allowed to create a book that offered different non-sequential navigation opportunities. The hypertext also enabled some early research on creating libraries that explained programming glossaries [1] and programming examples [13, 19] where non-linear navigation was most essential. By the middle 1990es, the process of making full-scale hypertext-based textbook has been mastered [16] and researchers in the area of computer science education started to explore some opportunities to extend hyper-textbooks with more advanced functionalities such as "runnable" program examples and simple problems [2, 22]. The progress, however was slow due to technical difficulties in working with traditional hypertext systems and interfacing them with other applications (such as compilers).

The gradual transition from classic hypertext to the Web supported a range of more advanced projects. In 1996, ELM-ART system [4, 5] pioneered an idea of an interactive and adaptive Web-based

[1] https://www.khanacademy.org/
[2] https://www.lynda.com/

[3] http://interactivepython.org/runestone/static/StudentCSP/index.html
[4] http://lti.cs.vt.edu/Books/OpenDSA/html/

Koli Calling 2017, November 16–19, 2017, Koli, Finland

ACM ISBN 78-1-4503-5301-4/17/11…$15.00
https://doi.org/https://doi.org/10.1145/3141880.3141890

textbook integrating an electronic textbook with an intelligent tutoring system that supported problem solving. This combination allowed creating textbooks with "live" LISP problems that students could solve online while receiving extensive feedback. The system also included "runnable" program examples and a function glossary. In 1997 Brown and Najork [3] followed with the model of "collaborative active" textbook that included dynamic examples of algorithm animations and collaboration functionality. By the end of 1990es, a few likeminded project explored other aspects of interactive Web-based electronic textbooks for computer science education [9, 10].

Just in a few years the work on interactive textbooks became an active research direction recognized by the first review of this area [23]. Several working group meetings [12, 18] helped to bring researchers in this area together and lead to some well-known collaborative projects such as OpenDSA [20]. Surprisingly, a gradual switch of interactive textbooks from being a focus of research to being practical tools resulted in decreasing attention to evaluating the impact of these novel tools. While many systems were developed (see [12, 18] for a good list of examples), very few studies have been reported. A notable recent example is the research by Ericson et al. [6–8] who explored the usage of different features and design principles of an e-textbook targeted for school teachers in programming education. Our current paper, on the other hand, attempts to bridge the gap between practice and research by focusing on evaluation of a typical modern interactive electronic programming textbook in a large university level course.

3 THE TARGET COURSE

This study was carried out in Fall 2015 and Fall 2016 on an introductory programming course at Aalto University, which is the largest technical university in Finland, and provides also programs in business, arts, and design. The target course was CS-A1111, Basic Course in Programming Y.1 This is a compulsory semester-long CS1 service course for the first year engineering students (not including computer science students) at the School of Engineering and at the School of Electrical Engineering. Most of these students take only 1–2 programming courses in their bachelor's studies. The programming language used in the course is Python. The course focuses on procedural programming covering all basic structures, like variables, selection statements, loops, functions, parameters, lists, and handling text files. Only the basics of the object-oriented programming are covered in the course, and it is possible to pass the course with a low grade without any knowledge of objects.

To pass the course, students had to pass the exam and get enough points from the mandatory exercises. The exercises were split to 9 weekly rounds, each having 3–4 programming problems. In addition, there were some program animations to explore and in 2015 also some Parsons problems [15]. Each round had a specified minimum number of points that students had to obtain to pass the course. However, it was possible to pass the course with a low grade without gathering minimum points from Round 9, which concerned object-oriented programming. In addition to the mandatory exercises, the students were offered an access to a voluntary practice system Python Grids exercises, which provided four types of interactive practice content: annotated examples, animated examples, semantic code assessment problems, and Parsons problems.

The course staff included the responsible teacher giving the lectures and 17 part time teaching assistants helping students to solve the exercise problems in classroom exercise sessions. In 2015, there were 17 and in 2016 18 weekly exercise groups during 10 weeks. Attendance at the exercise sessions was voluntary. All exercise problems were submitted through a web page to an automatic assessment tool, thus allowing students to work from home or elsewhere off campus. On the other hand, students could take part in several exercise sessions a week, if they wanted more help. The course had also a web forum where the students could ask for advice.

The contents and requirements of the course in 2015 and 2016 were the same. The main difference was that in 2015, there were 18 lectures together covering all course topics, and the main learning resource was a static PDF textbook [5] written by the lecturer. In 2016, the number of the lectures was reduced to 10, the lectures concentrated on basic concepts, and the students were asked to self-study the more advanced examples using an interactive e-textbook [6] which was developed from the previous year static resource. The text in both books was almost the same. However, the e-textbook also included interactive animations, using technology developed in [21], and annotated code examples. Moreover, in 2016 there was also a direct link from the exercises to the corresponding e-textbook chapter while in 2015, no such link existed, and there was only the whole PDF book available as a resource.

An example of an animation is presented in Figure 1. The animation shows the program code, the line of the code which is currently executed, the expression to be evaluated, the current values of the variables and so on. Animations also contain manually added explanations for the key steps.

An example of an annotated code example is presented in Figure 2. The PDF book included nearly the same code examples, but the code was explained in plain text, while in the annotated code examples the explanation is split into small parts presented in boxes. The reader can move the cursor on a certain box, and the code line(s) which are explained in this box are colored with a red color.

4 THE STUDY

As described above, the course arrangements, requirements and staff resources were very close to each other in the 2015 and 2016. The role of lectures was diminished, increasing the need for self studying. On the other hand, changing the static PDF textbook into an interactive e-textbook and setting up direct links from the assignments to the corresponding e-textbook chapters were expected to support self studying.

We wanted to explore whether these differences had an effect on students' motivation factors and learning results. Our research questions are: *RQ1) Were there differences in the changes of the motivation factors of the students in 2015 and 2016 courses?, RQ2) Were there differences in the learning gain (the difference of the posttest and pretest results) in 2015 and 2016 courses?, RQ3) Were there differences in the exam results of 2015 and 2016 courses?*

[5] http://www.cse.hut.fi/fi/opinnot/CSE-A1111/S2015/kalvot/opetusmoniste2015.pdf
[6] https://grader.cs.hut.fi/static/y1/

Figure 1: Example of an animation in e-textbook.

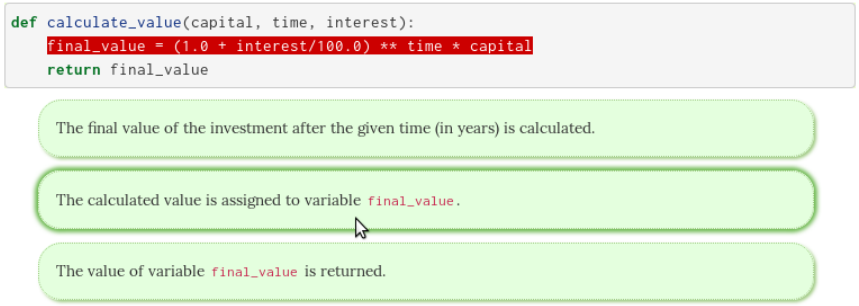

Figure 2: Example of an annotated example in e-textbook. (original Finnish example translated into English).

4.1 Data collection

In both years, the students were asked to take a pretest on Python programming at the beginning of the course and posttest at the end of the course. Each test contained 10 small Python problems. Posttest problems were isomorphic to the pretest. The tests were the same in 2015 and 2016.

The students also completed a Motivational Questionnaire at the beginning and at the end of the course. This questionnaire contains two instruments: the Learning Activation questionnaire and the Achievement-Goal Orientation questionnaire. The Learning Activation questionnaire has been adapted from the questionnare developed by [14] and it includes three motivational factors: *Fascination* (4 questions), *Competency Beliefs* (5 questions), and *Values* (5 questions). Items of Fascination factor measure the extent to which the student likes programming. Competency Beliefs questions ask students if they think they can deal positively with the subject. Values questions measure to which extent students think programming is important for their lives and professional development.

The Achievement-Goal questionnaire is a 12-question survey which measures Goal-Orientation conformed of 4 factors which are not exlusive: *Mastery Approach* orientation is related to the motivation of mastering the learning content; *Mastery Avoidance* is related to the avoidance of failing to learn; *Performance Approach* relates to the motivation to perform, score, or do better than others; and *Performance Avoidance* orientation refers to the motivation for avoiding to fail, scoring under the minimum, or doing worse than others.

In the end of the course, the students took a final exam, which was compulsory to pass the course. The exam consisted of small problems where the student had to explain what a given program outputs or the purpose of the given function (total 21 % of maximum points), and problems where the student had to write a small Python program with pen and paper (total 75 % of maximum points). 4 % of maximum points were given as a bonus if the student took the posttest and the final motivation query. The problems in 2015 and 2016 exams were different, but they tested the same concepts.

In the end of the course, the students were asked to fill a feedback questionnaire, which included 50 questions, including questions about usage of the course material and their opinions on it.

4.2 Results

Regarding motivation, 426 and 444 students answered both initial and final motivational questionnaires in 2015 and 2016, respectively. For each student, answers of questions within each motivational factor were averaged and normalized to the range 0 to 1. A value

of 0 is the lower motivation, and the value of 1 is the maximum motivation. Table 1 shows the average values of the factors at the beginning and at the end of each course and their differences. Non-parametric Mann-Whitney did not show differences between years in the initial measure of any motivational factor, except for Competency Beliefs (p=.021), being this motivational factor lower in 2016. From Table 1 we observe differences in the changes of motivation factors from the initial motivation questionnaire to the final motivation questionnaire. To test these differences we build regression models in which the year is added as a dummy variable, with value 0 for 2015 and 1 for 2016. Models were built separately for each motivational factor measured at the end of the term (final) and included as predictors the motivational factor at the beginning of the term (initial) and the dummy variable *year*. We report the models in which *year* contributes significantly to improve the a preliminary model built only with the motivational *initial* measure as predictor. The Beta coefficient of *year* and its significant value is reported in last column of Table 1.

The Competency Beliefs was lower at the beginning of the 2016 course, but it increased more than in 2015 course. Also, in 2016, the increase of Values is significantly higher than in 2015. A positive effect is also observed for Mastery approach. While in both groups this motivational factor decreases, it decreased less in 2016. In summary, the motivational factors developed more into positive direction in 2016 than in 2015.

The summary of the results of the pretest and the posttest are presented in Table 2. The pretest scores were slightly higher in 2015 than 2016 (ie. the students' initial knowledge was slightly better in 2015), but the posttest results were clearly better in 2016 than 2015. From the pretest and posttest results we calculated the *learning gain* for each student. It is defined as a ratio of the actual gain (posttest score minus pretest score) to the maximum possible gain (maximum achievable posttest score minus pretest score). In 2015, the average learning gain was 0.466, while in 2016, it was 0.526. Thus, the learning gain was 0.06 larger in 2016 than in 2015. To test for significance, we built first a regression model in which posttest is predicted first by pretest, and then a second model that adds the dummy variable *year*. Results showed a significant contribution of *year*, i.e a positive effect of using the e-textbook, in predicting posttest, $\beta_{year} = .071, p < .000$), which can be interpreted as using e-textbook explains an increase of 7% in posttest scores, controlling for pretest scores.

There was a small difference in the results of the first exams of the course in 2015 and 2016. The maximum number of points in the exam was 100. 4 of those points were given as bonus points if the student took the posttest and answered to the final motivation questionnaire. In both years, 556 students took part in the exam. In 2015, the average of the exam points was 79.38, while in 2016, the average was 77.26. Thus, the average was 2.12 points lower in 2016. However, difference of 0.74 points comes from the bonus points (average was 3.99 in 2015 while it was 3.25 in 2016). Thus, the real difference in actual exam problems was 1.38 points. Because the problems were different in various years (although they were about the same topics), we cannot, however, conclude whether the difference is due to the different problems or students' knowledge.

According to the feedback questionnaire at the end of the course, more students really used the e-textbook as a course material in 2016 than PDF book in 2015. In 2015, 63.0 % of the students told that they had used the PDF book, while in 2016 92.6 % of the students told that they had used the e-textbook. It is rather strange that part of the students did not read the PDF book or e-textbook, because it had been told to be the main course text in the course. However, we have no data to explain this. We can speculate that these students used other online or printed resources.

In addition to the change of the format of the book, the part of the increase in the use might result from the fact that in 2016, most of the Python programming problems had at the beginning of the problem description a direct link to the chapter of the e-textbook which told about the programming concepts needed in solving the problem. (For example, if it was necessary to use lists to solve the problem, the problem description had a link to a e-textbook chapter about lists.) In 2015, is was told at the beginning of the problem description which topics are practiced in the problem, but no direct links were provided. The static PDF text was accessible or downloadable in a fixed place in the online learning environment.

The students were also asked to evaluate the PDF book in 2015 and the e-textbook in 2016 using a grade in scale 1–5, where 5 is the best. Table 3 shows these results along with the number of students who declared not using the material in both years. Note that fewer people missed to use the e-textbook (20) than students who did not use the PDF version in 2015 (118). In 2015, the average of the grade was 3.68, while in 2016, the average was 3.97. Non-parametric Mann-Whitney test shows a significant difference, Mann-Whitney $U = 53687.0$, $p < .000$, $MeanRank_{2015} = 329.66$, $MeanRank_{2016} = 406.49$, showing higher values in 2016. Thus, the students liked the e-textbook better than the PDF book, although the text was almost the same in both versions.

These differences cannot be explained by the differences of the students, because the students came from the same degree programs in 2015 and 2016, and there were no significant differences in the previous knowledge of the students according to the pretest results.

5 DISCUSSION

Students' activity and engagement with learning resources is in a central role in the learning process. Previous research has shown that engagement with interactive materials, e.g., [11] is beneficial for learning. Interaction with immediate feedback provides students a valuable resource for exploring new concepts, as well as testing their mental models about the concepts and their behavior. The feedback supports correcting and tuning their mental models thus supporting learning. This is the strength of interactive e-textbooks compared with static resources.

However, usage of e-textbooks is a complex phenomenon, and students are not using them in isolation. Our current research is a pilot study concerning the usage of an e-textbook. While we observed several positive results both in motivation and learning results, we should treat these results with care. There were changes in the lectures as well as weekly programming exercises, which can include intervening variables which have an effect in the results. It is plausible that the decrease of lectures encouraged students for self studying the material. However, only a minority of students have followed the lectures to the end in both years, which implies that this effect might be small. On the other hand, the clear increase

Table 1: Summary of the results of motivation queries.

	2015			2016			regression
Factor	initial	final	difference	initial	final	difference	$\beta_{initial}$ (p-value)
Fascination	.565	.577	.012	.578	.596	.018	
Competency Beliefs	.504	.656	.152	.485	.681	.197	.032 (.006)
Values	.702	.684	-.017	.707	.727	.020	.032 (<.000)
Mastery Approach	.693	.642	-.051	.699	.674	-.025	.032 (.005)
Mastery Avoidance	.617	.578	-.039	.619	.601	-.018	
Performance Approach	.574	.568	-.006	.575	.588	.014	
Performance Avoidance	.571	.552	-.019	.575	.567	-.008	

Table 2: Summary of the pretest and posttest results.

	2015		2016	
	N	Mean	N	Mean
pretest	532	0.2221	545	0.2019
posttest	429	0.6018	459	0.6634
learning gain	412	0.4662	450	0.5263

Table 3: Feedback given by students to the PDF material in 2015 and e-textbook in 2016.

	Answers					Did not	
	1	2	3	4	5	use	Total
2015	0%	6%	36%	43%	15%	118	427
2016	0%	5%	20%	47%	28%	20	479

of using the e-textbook, covering almost the whole 2016 student cohort, suggests that students considered the resource useful and used it when working with the exercises.

For future research we need to log more accurately how students accessed the chapters and how much they used the interactive elements in the book. Some interesting questions include further exploring the difference in impact of using static PDFs vs. e-textbooks with interactive elements: what is the value (impact) of integrating interactive learning resources into an e-textbook vs. using only links to external interactive elements, what is the impact of different types of interactive elements in an e-textbook, when considered separately or as a combined resource, and how do different types of students use these resources?

REFERENCES

[1] H. D. Böcker, H. Hohl, and T. Schwab. Hypadapter - Individualizing Hypertext. In D. Diaper, editor, *IFIP TC13 Third International Conference on Human-Computer Interaction*, pages 931–936. North-Holland, 1990.

[2] T. Boyle, G. Gray, B. Wendl, and M. Davies. Taking the plunge with CLEM: the design and evaluation of a large scale CAL system. *Computers and Education*, 22(1/2):19–26, 1994.

[3] M. H. Brown and M. A. Najork. Collaborative active textbooks. *Journal of Visual Languages and Computing*, 8(4):453–486, 1997.

[4] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An intelligent tutoring system on World Wide Web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Third International Conference on Intelligent Tutoring Systems, ITS-96*, volume 1086 of *Lecture Notes in Computer Science*, pages 261–269. Springer Verlag, 1996.

[5] P. Brusilovsky, E. Schwarz, and G. Weber. *Electronic textbooks on WWW: from static hypertext to interactivity and adaptivity*, pages 255–261. Educational Technology Publications, Englewood Cliffs, New Jersey, 1997.

[6] B. Ericson, S. Moore, B. Morrison, and M. Guzdial. Usability and Usage of Interactive Features in an Online Ebook for CS Teachers. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, pages 111–120, New York, NY, USA, 2015. ACM.

[7] B. J. Ericson, M. J. Guzdial, and B. 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*, ICER '15, pages 169–178, New York, NY, USA, 2015. ACM.

[8] B. J. Ericson, K. Rogers, M. C. Parker, B. B. Morrison, and M. Guzdial. Identifying Design Principles for CS Teacher Ebooks through Design-Based Research. In *Proceedings of the 12th Annual International Conference on International Computing Education Research*, ICER '16, pages 191–200, 2016.

[9] O. Grillmeyer. An Interactive Multimedia Textbook for Introductory Computer Science. In *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, pages 286–290, New York, NY, USA, 1999. ACM.

[10] N. Henze, K. Naceur, W. Nejdl, and M. Wolpers. Adaptive hyperbooks for constructivist teaching. *Künstliche Intelligenz*, 13(4):26–31, 1999.

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

[12] A. Korhonen, T. Naps, C. Boisvert, P. Crescenzi, V. Karavirta, L. Mannila, B. Miller, B. Morrison, S. Rodger, R. Ross, and C. 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*, pages 53–72. ACM, 2013.

[13] M. Linn. How can hypermedia tools help teach programming. *Learning and Instruction*, 2:119–139, 1992.

[14] D. W. Moore, M. E. Bathgate, J. Chung, and M. A. Cannady. Measuring activation and engagement. Technical report, Activation Lab, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, PA (USA), 2011.

[15] 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.

[16] R. Rada. Converting a textbook to hypertext. *ACM Transactions on Information Systems*, 10(3):294–315, 1992.

[17] J. R. Remde, L. M. Gomez, and T. K. Landauer. SuperBook: an automatic tool for information exploration – hypertext? In *the ACM conference on Hypertext, Hypertext'87*, pages 175–188, 1987.

[18] G. Rößling, T. Naps, M. Hall, V. Karavirta, A. Kerren, C. Leska, A. Moreno, R. Oechsle, S. Rodger, J. Urquiza-Fuentes, and J. A. Velázquez-Iturbide. Merging Interactive Visualizations with Hypertextbooks and Course Management. *SIGCSE Bull.*, 38(4):166–181, 2006.

[19] P. K. Schank, M. C. Linn, and M. J. Clancy. Supporting PASCAL programming with an on-line template library and case studies. *International Journal on the Man-Machine Studies*, 38(6):1031–1048, 1993.

[20] C. Shaffer. OpenDSA: An Interactive eTextbook for Computer Science Courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 5–5. ACM, 2016.

[21] T. Sirkiä. Jsvee & Kelmu: Creating and Tailoring Program Animations for Computing Education. In *2016 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 36–45, Oct 2016.

[22] S. Tyerman, P. Woods, and J. Warren. Loop Tutor and HyperTutor: Experiences with adaptive tutoring systems. In *Proc. of Australian and New Zealand Conference on Intelligent Information Systems, ANZIIS'96*, pages 60–63, 1996.

[23] M. Unanue, P. Velasco, P. Flores, U. Fuentes, and I. Velázquez. Electronic books for programming education: a review and future prospects. In *Proceedings of the 7th annual conference on Innovation and technology in computer science education, ITiCSE '02*, pages 34–38, 2002.