To Code or Not to Code? Programming in Introductory CS Courses†

Jennifer Parham-Mocello
School of EECS
Oregon State University
Corvallis, USA
parhammj@oregonstate.edu

Martin Erwig
School of EECS
Oregon State University
Corvallis, USA
erwig@oregonstate.edu

Emily Dominguez
School of EECS
Oregon State University
Corvallis, USA
domingue@oregonstate.edu

***Abstract*— Code-first approaches for introducing students to CS exclude those without preparatory privilege in programming and those intimidated by coding. Delaying coding or not using coding in an introductory CS course provides an equitable learning opportunity and includes a broader group of students in computational education. We present a study that compares a traditional Python code-first approach with an approach to delay or remove coding by first using simple, well-known stories to explain computation without the need for a computer or coding. We find that many students, especially female students and those without prior programming, are initially not interested in coding but in using stories to explain computing. We conclude that a traditional Python code-first approach excludes these students and an option using stories is a viable alternative.**

Keywords— Story Telling; Computer Science Pedagogy; Introduction to Computer Science

I. INTRODUCTION

Coding can be frustrating to students [13, 22], and thus an immediate focus on coding in computer science education can discourage students from considering computer science (CS) as a field of study. In addition, coding-based teaching approaches create a divide between those who *have* prior programming experience and those who *have not*, which leads to feelings of low self-efficacy, inferiority, and attrition among females [15] and underrepresented minorities [16].

Thus, many current coding-based approaches to the introduction of computer science and computational thinking (CT) create inequitable entrances into CS education by inhibiting participation and suffer from three sources of inequity affecting different groups of students:

- *Intimidation* (affects those who are put off by coding),
- *Preparatory Privilege* (affects those who have not had prior programming experience), and
- *Race & Gender*

These factors are not completely independent, but they do cover a larger group of students than any single one. For example, white males can be intimidated or not have prior programming experience.

Programming predicates many approaches introducing computer science to students; that is, they require an understanding of how to code an algorithm in a programming language. *Code.org* uses this approach to promote CS to younger children, but the abstract nature of programming languages (block based or not) can pose a significant barrier to entry. A code-first approach can be effective when students have a good understanding of programming or are willing to acquire it. However, since computer science is not synonymous with programming, there is no inherent necessity to tie the orientation to computer science to coding activities.

We believe that the state-of-the-art introductory computer science education at the university level could benefit from more creativity and computational thinking without the use of a computer. To reach a wide audience with a diverse background and set of expectations, we employ the so-called *Story Programming* approach that uses well-known stories and everyday situations to explain computer science concepts before teaching coding. This approach is based on the book [6] and was first proposed in our previous work [18] where we found that delaying coding by 5 weeks and using stories to explain computation is a viable approach compared to a traditional code-first approach to teaching a CS orientation course (CS 0). Delaying coding bears a number of risks, including frustrating students who are expecting to learn how to code and who fear that they might get behind compared to others who start in a coding-first environment. To explore the non-coding approach further, we decided to take an even bigger risk and remove coding altogether. In the remainder of this paper we outline the approach and present a comparison of the traditional code-first with the story approach that removes or delays coding.

We find that there are a significant number of students, especially females and those without prior programming experience, who are initially not interested in coding but in using stories to explain computing. We conclude that offering a traditional Python code-first approach excludes these students and an option using stories to explain computing is a viable alternative

II. BACKGROUND

The two most closely related areas to our approach are the so-called "unplugged" computational thinking approach and other approaches to using stories for explaining computing concepts.

A. Unplugged CT activities

The most well-known approach to teaching CS concepts without the use of a computer is the approach taken in *csunplugged.org* [3]. It is a collection of engaging activities that can illustrate computing concepts, but it does not use stories. The Story

† This work is partially supported by the National Science Foundation under the grant CCF-1717300.

978-1-7281-0810-0/19/$31.00 ©2019 IEEE

Programming approach uses the idea of unplugged activities performed without a computer to introduce the computational concept before coding, but the actual non-coding activities relate to the existing stories or stories students create.

Thus far the unplugged activities have been primarily employed in the K-12 education [1, 4, 9, 19, 20]. The research presented in this paper investigates the idea of teaching computational thinking without a computer at the university level. Most alternatives for teaching introductory CS courses focus on changing the curriculum to improve success and retention [7, 17] and make topics covered more relevant and broader [14]. Some universities create interest-based classes allowing students to choose a class section based on what they like, such as game development, robotics, music, and mobile applications [7, 23], while others focus on adding computational thinking to their curricula with and without the use of a computer [8, 11, 17, 21]. These studies show that adding computational thinking to a curriculum helps students think about different ways to attack problems and makes them more effective problem solvers. The Story Programming approach uses interest in stories as a tool for teaching computational thinking in CS 0.

B. Approaches Based on Stories

There are other approaches using stories to explain computing. For example, the author of [12] describes a story about a princess on a quest to save her father's kingdom and introduces algorithms and data structures along the way. The target audience is middle school children. The author of [2] employs a similar approach and tells a story about a girl who wants to find her way back home after getting lost in a forest. As part of her quest, she has to solve several problems, which are used to introduce concepts of algorithms and math. The story is a bit like *Alice in Wonderland* with its playful and clever use of names, and the target audience is again middle school children.

One study introduced *Computational Fairy Tales* alongside coding to help the retention and academic performance of CS majors, mostly aimed at students with little to no programming experience [17]. It found that "CS0 students without prior programming experience got significantly higher grades in CS1 than CS0 students who had programmed before"; the students were split on how useful the book was to their learning. This is different than the study presented in this paper, which removes coding from the Story Programming approach and compares it with a traditional Python code-first approach. Another difference is that Story Programming employs existing stories students are already familiar with, which means students don't have to absorb a new story before making the connection to computational concepts.

Another study claimed that using a story to learn a concept will be easily accessible because that is how many of us learn to begin with [10]. These claims align with the rationale for using a Story Programming approach, but our study presented in the following sections does not investigate these claims. One other study used unplugged activities and storytelling to introduce teachers to computational thinking, but it focused on teacher training and used contextual stories to relate different unplugged activities to specific computational skills for teachers as the storytelling approach [5].

Most closely related to the study reported in this paper is our previous work [18], which describes an experiment of varying an existing computers science orientation course. This course is primarily taken by incoming first-year students declared as CS majors, although upper-level CS students or those outside the major may take the course as well. Traditionally, the course taught Python as a coding language with students beginning to write small programs as early as week two in a ten-week term. In the prior study from fall 2017, the course was divided into three sections. Two of the sections introduced a new so-called *Story Programming* approach, which is based on the book [6] using stories to explain computation, and delayed coding by 5 weeks. The other section remained the same as traditionally taught in previous years. One of the two Story Programming sections used Python in the second half of the class, while the other section used Haskell. The goal was to investigate the new approach and differences in language choice with the new approach. Also, the students were not told about the different approaches prior to the class to elicit an unbiased opinion of the approach from students who do not self-select based on interest. The most important result of that study was that a delayed coding approach is a viable alternative to the coding-first approach.

III. Research Method

The study described in this paper pushes the idea of delayed coding further and investigates how well a "Story No Code" approach, which doesn't use coding at all, might work. We decided to also offer a "traditional" Python coding section that uses the same curriculum as the Python section from [18] to obtain a direct comparison. To avoid some of the student frustration about lack of coding that was reported in [18], we decided to let students self-select into the sections based on a small description of the two different sections. The same instructor who taught the fall 2017 courses also taught the two sections in fall 2018.

In the following, we describe the structure of the course sections from [18] and our study and the research questions. We have kept the curriculum of our Story Programming section (which has essentially become a Story No Code section) as close as possible to the Story Programming sections from [18] to facilitate a meaningful comparison.

A. Course Structure

The Story Programming sections from fall 2017 and 2018 had between 65 and 105 students per lecture and used presentation slides to teach concepts, as well as live coding demonstrations through a terminal when teaching programming in the two delayed-coding sections. The students in the Story Programming sections read relevant chapters of the book before class with a weekly online quiz. Every week students engaged in in-lecture group exercises focused on understanding the stories used in the book from a computational perspective, coming up with new stories to explain computational concepts, and developing and tracing algorithms.

All sections had two one-hour lectures each week and one two-hour lab per week, and the first 5 weeks of all the Story Programming labs focused on small group activities applying concepts to the real world. For example, one of the first activities

examined path finding algorithms using the tale of Hansel and Gretel, and this activity presented the students three variants of the algorithm that they had to act out with pebbles. Another activity helped students learn about the runtime of different algorithms by having to count and transfer beans across the classroom using different methods. In the delayed-coding sections, the programming activities in the last 5 labs and assignments used code to mirror the concepts covered in the first 5 weeks, rather the new concepts covered in class; whereas, the non-coding section continued using hands-on, small group activities related to the concepts covered in lecture and the book.

B. Research Questions

Offering a new, non-coding Story Programming section that students self-selected into allowed us to investigate the effects of removing coding from a university CS orientation course and explore who is interested in the approach using stories versus Python programming before the class.

- *RQ1: Do the DWF (Drop/Withdraw/Failure) rates and grade distributions differ when coding is removed?*
- *RQ2: What are students' initial interests in the class, coding, and the use of stories to explain computing, and do these interests change after the class?*

C. Data Collection

With IRB permission, we collected course-level DWF rates and grade distribution information from the registrar, and we collected student-level pre- and post-survey data from consenting participants.

IV. Results and Discussion

The course-level results for RQ1 are out of 191 students, and the student-level survey results for RQ2 are out of 147 consenting participants (73 Story No Code and 74 Fall18 Python). Since the data are not normally distributed and are based on a Likert scale (ordinal), we use non-parametric statistical tests, i.e., the Kruskal-Wallis hypothesis test for 2 or more categories and the Wilcoxon paired t-test between dependent pre and post information to reject hypotheses with 95% confidence, $\alpha <= .05$.

RQ1: Do the DWF rates and grade distributions differ when coding is removed?

Does removing coding retain more students and increase their grade point average compared to the code-first approach traditionally used to teach the CS Orientation course? In fall 2017 we observed that using the Story Programming approach and delaying coding, regardless of language, did not retain (nor did it lose) more students than the traditional Python code-first approach (6.5%-9.2%) [18]. In fall 2018 we observe the same thing when coding is completely removed. The DWF rates using different approaches do not significantly differ, but it is interesting to note that the DWF rate doubles from 6.5% in fall 2017 to 13.7% in fall 2018 in the traditional code-first section with the same exact curriculum and instructor. Overall, the fall 2018 sections had a higher DWF rate than the fall 2017 section, which suggests the academic year has more to do with the DWF rates than the approach used in this class.

When coding is removed, students do not get a significantly higher grade-point average than when coding is delayed or used immediately. However, there is a significant difference in the Story Programming approach delaying coding with Python and the grades in the fall 2018 code-first traditional Python programming approach. Across all sections, the majority of the students are As (60-82%), with only 2-5% of the students making Cs. However, the Story Python with delayed-programming had significantly more As than the fall 2018 traditional Python code-first approach, which has more Bs and higher DWF rates. Overall, the approach using stories without coding or delaying coding seems to yield more consistent DWF rates and grade distributions than the traditional Python code-first approach.

RQ2: What are students' initial interests in the class, coding, and the use of stories to explain computing, and do these interests change after the class?

Since students self-selected into the different sections in fall 2018, we asked questions about their interests in the class, coding, and use of stories to explain computing before and after the class to understand (1) who and what are the interests in each of these areas before entering the course and (2) if there are any changes in these interests after the class.

1) What are the initial student interests in the class and learning more about programming/coding, and do these interests change after the class?

Prior to the courses, there are significant differences in students' interest in the class and learning more about programming/coding among the two sections with and without coding. More students in the Python programming section are extremely interested in the class than in the Story No Code section, which could be because they are in the section for a reason other than interests or because they are unsure of a new approach using stories without coding (see Fig. 1). It makes sense there are less students extremely interested in coding in the Story No Code section (see Fig. 1), but overall, coding extremely interests most students in both sections. This is likely due to students self-selecting to take a computer science orientation course and having interest in majoring in CS, but it is worthwhile pointing out that almost 20% of the students in the Story No Code section were only somewhat interested in coding, which means that a code-focused orientation to CS does not necessarily appeal to all students and may be worse for non-majors.

A paired t-test of pre- and post interests in the class shows a significant decrease in interests in the class within the Python coding section and the Story No Code section, with Python having a larger decrease. The change in student interests in both classes could be because the course content did not meet students' expectations or because the students did not sign up for this section based on interest but rather for scheduling reasons or availability in the class. However, an unexpected result is that there is a significant decrease in students' interest to learn more about coding after taking the Python coding class; whereas, the students' interest in learning more about coding does not change at all in the Story No Code section (see Fig. 1).

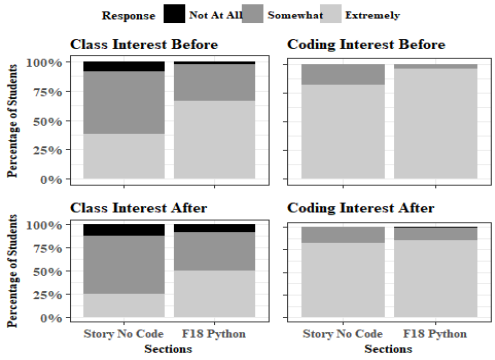

Fig. 1. Distribution of student responses to survey question: "Rate your interest in… 1) this class and 2) learning more about programming/coding".

2) How would using stories to explain computation versus writing programs/code affect student interests in the class and motivation to learn more about CS and coding, and do these interests change after the class?

We notice that writing programs/code interest students more than using stories to explain computing (see Fig. 2), and coding and using stories does not significantly differ among sections. However, it is worth mentioning that more students in the Python section said that having stories would increase motivation to learn about CS than the Story No Code section, which is likely due to students in sections for reasons other than interest. Even though there is more interest and motivation for writing code, 40-45% of the students say that the use of stories would greatly or slightly increase their interests in the class and motivation to learn more about CS and coding (see Fig. 2), which suggests that a combination of the story approach with coding might be better.

Across sections, there is a significant difference in students' pre- and post-response to how using stories to explain computing would (or did) affect their interest in the class and motivation to learn more about CS and programming/coding. In the Python section, students responded more negatively about the use of stories after the class; whereas, the students in the Story No Code section responded more positively about the use of stories after the class. It is interesting to note that we see fewer positive responses about the effect writing programs/code had on interests in the class and motivation to learn more about CS before and after the class with coding.

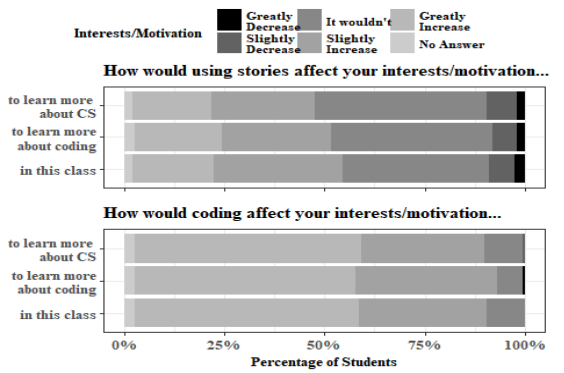

Fig. 2. Distribution of student responses to a pre-survey question: "How would using stories to explain computing affect your interest/motivation …" and "How would writing programs/code affect your interest/motivation …"

3) Who is initially interested in the class, coding, and using stories to explain computing?

To better understand who is interested in the different approaches, we look at whether prior programming or gender play a role in students' initial interests. There is no difference in coding interests based on the gender or prior programming, but prior programming experience and gender do play role in the interest of the class and the use of stories to explain computing.

There is no significant difference between students' with and without prior programming and their initial class interests in each section. However, it is interesting to note that those without prior programming are more interested in either class than those with experience, and students with prior programming are the only ones who ever respond that they are not interested in the class at all or are more likely to only be somewhat interested, which suggests that these students are not interested in CS 0. Overall, there are a significant number of students without prior programming experience who say the use of stories to explain computing would greatly increase their interest in the class and motivation to learn more about coding and CS before the course. This could be because students without prior programming experience want a non-coding or different approach to learning about CS than those who already chose to get into CS by programming.

There is no difference in female and male interests in the Python coding class, but there is in the Story No Code approach. In this section, females are intitially more interested in the class, which suggests that a non-coding or story approach might be more attractive to females. It is interesting to note that only male students responded not-at-all-interested in the class. While there are differences in responses to all questions about using stories and students with and without prior programming experience, there are not as many differences between responses from different genders. However, more females initially stated that using stories to explain computing would greatly increase their motivation to learn more about CS.

V. Conclusions

Based on the results from this study, we conclude a code-first approach for CS 0 excludes some students and is advantageous to those students to continue offering the story approach as an alternative for the following reasons:

- The approach and amount of coding used to orient students to CS has less of an effect on DWF and grades than the term in which a class is taught.
- A significant number of students think the use of stories to explain computing would increase their interest in the class and motivation to learn more about CS or coding
- Those without prior programming experience and females are initially more interested in and motivated by the non-coding class using stories. Almost 20% of the students are only somewhat interested in coding.
- A code-first approach negatively impacts interests in coding and learning more about CS.

REFERENCES

[1] T. Bell, J. Alexander, I. Freeman, and M. Grimley. 2009. Computer Science Unplugged: school students doing real computing without computers. *Journal of Applied Computing and Information Technology 13*, 1.

[2] C. Bueno. 2014. *Loren Ipsum*. No Starch Press.

[3] CS Education Research Group. CS unplugged: Computer Science without a computer. http://www.csunplugged.org

[4] P. Curzon. 2013. cs4fn and computational thinking unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (WiPSE '13). ACM, New York, NY, USA, 47-50.

[5] P. Curzon, P. W. McOwan, N. Plant, and L. R. Meagher. 2014. Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (WiPSCE '14). ACM, New York, NY, USA, 89-92.

[6] M. Erwig. 2017. *Once Upon an Algorithm: How Stories Explain Computing*. MIT Press.

[7] M. Haungs, C. Clark, J. Clements, and D. Janzen. 2012. Improving first-year success and retention through interest-based CS0 courses. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 589-594.

[8] P. B. Henderson. 2011. Computing unplugged enrichment. *ACM Inroads 2*, 3 (August 2011), 24-25. DOI: http://dx.doi.org/10.1145/2003616.2003626

[9] F. Hermans and E. Aivaloglou. 2017. To Scratch or not to Scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (WiPSCE '17), Erik Barendsen and Peter Hubwieser (Eds.). ACM, New York, NY, USA, 49-56.

[10] W. J. Joel. 2013. A story paradigm for computer science education. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (ITiCSE '13). ACM, New York, NY, USA, 362-362.

[11] D. Kafura and Deborah Tatar. 2011. Initial experience with a computational thinking course for computer science students. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 251-256.

[12] J. Kubica. 2012. *Computational Fairy Tales*. CreateSpace Independent Publishing Platform.

[13] A. Lishinski, A. Yadav, and R. Enbody. 2017. Students' Emotional Reactions to Programming Projects in Introduction to Programming: Measurement Approach and Influence on Learning Outcomes. *ACM Conf. on International Computing Education Research (ICER 2017)*, pp. 30-38.

[14] D. J. Malan. 2010. Reinventing CS50. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (SIGCSE '10). ACM, New York, NY, USA, 152-156

[15] J. Margolis and A. Fisher. 2003. *Unlocking the clubhouse: Women in computing*. Cambridge, Mass: MIT Press.

[16] J. Margolis. 2008. *Stuck in the shallow end education, race, and computing*. MIT Press.

[17] C. Marling and D. Juedes. 2016. CS0 for Computer Science Majors at Ohio University. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16). ACM, New York, NY, USA, 138-143

[18] J. Parham-Mocello, S. Ernst, M. Erwig, L. Shellhammer, and E. Dominguez. 2019. Story Programming: Explaining Computer Science Before Coding. *SIGCSE '19*, February 27-March 2, 2019, Minneapolis, MN.

[19] B. Rodriguez, C. Rader, and T. Camp. 2016. Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '16). ACM, New York, NY, USA, 95-100.

[20] R. Thies and J. Vahrenhold. 2013. On plugging "unplugged" into CS classes. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 365-370. .

[21] M. Van Dyne and J. Braun. 2014. Effectiveness of a computational thinking (CS0) course on student analytical skills. In *Proceedings of the 45th ACM technical symposium on Computer science education* (SIGCSE '14). ACM, New York, NY, USA, 133-138. .

[22] B. C. Wilson and S. Shrock. 2001. Contributing to success in an introductory computer science course: A study of twelve factors. *Thirty-Second SIGCSE Technical Symposium on Computer Science Education*

[23] Z. J. Wood, J. Clements, Z. Peterson, D. Janzen, H. Smith, M. Haungs, J. Workman, J. Bellardo, and B. DeBruhl. 2018. Mixed Approaches to CS0: Exploring Topic and Pedagogy Variance after Six Years of CS0. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (SIGCSE '18). ACM, New York, NY, USA, 20-25