

The Journal of Computing Sciences in Colleges

Papers of the 35th Annual CCSC
South Central Conference

April 5th, 2024
Stephen F. Austin State University
Nacogdoches, TX

Bin Peng, Associate Editor
Park University

Binyang Wei, Regional Editor
Texas Christian University
Mustafa Al-Lail, Regional Editor
Texas A&M International University

Volume 39, Number 7

April 2024

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors	5
CCSC National Partners	7
Welcome to the 2024 CCSC South Central Conference	8
Regional Committees — 2024 CCSC South Central Region	9
Reviewers — 2024 CCSC South Central Conference	10
The Evolution of IT in Higher Education and Preparing for the Future	
— Opening Keynote	11
<i>Michael Coffee, Stephen F. Austin State University</i>	
A Case Study on Adopting Best Practices in Introductory Computer Science	12
<i>Jeremy Becnel, Stephen F. Austin State University</i>	
Fostering Code Quality Practices Among Undergraduate Novice Programmers	21
<i>Essa Imhmed, Edgar Ceh-Varela, Scott Kilgore, Eastern New Mexico University; Hashim Abu-Gellban, Grand Canyon University</i>	
A Mobile App Leveraging NLP Techniques for Sci-Fi Book Recommendations	33
<i>Edgar Ceh-Varela, Essa Imhmed, Drey Smith, Eastern New Mexico University</i>	
Teaching Cross-Platform Mobile Development and Cultivating Self-Directed Learners – A Six-Week Summer Online Course Experience	41
<i>Liqiang Zhang, Indiana University South Bend</i>	
Hack the Border: Empowering Experiential Learning Competencies in Computing through Hackathons	52
<i>Christian Servin, Nadia Karichev, El Paso Community College; J.J. Childress, Microsoft</i>	

Designing a Design-Oriented Course for CS Majors	58
<i>Fahmida Hamid, New College of Florida</i>	
FpTracker—A Labware for Teaching Browser Fingerprinting and Privacy Preservation	64
<i>Lin Li, Na Li, Prairie View A&M University</i>	
Camp CryptoBot: A Model for Taking Risks and Promoting Self-Efficacy in Pursuit of Cybersecurity Career Pathways	72
<i>Pauline Mosley, Li-Chiou Chen, Lisa Ellrodt, and Doris Ulysse, Pace University</i>	
The Utility of Radix Representations and Surrogate Logarithms in the Analysis of Algorithms and Data Structures	82
<i>Michael Kart, Saint Edward's University</i>	
Learning Parallelism Through an Unplugged Class Activity — Conference Workshop	91
<i>Matthew Troups, Tulane University; Anurag Dasgupta, Valdosta State University; Venkat Margapuri, Villanova University; Simon Shamoun, Hofstra University; Shubhi Taneja, Worcester Polytechnic Institute</i>	

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Scott Sigman, President (2024),
ssigman@drury.edu, Mathematics and
Computer Science Department, Drury
University, Springfield, MO 65802.

Bryan Dixon, Vice
President/President-Elect (2024),
bcdixon@csuchico.edu, Computer
Science Department, California State
University Chico, Chico, CA 95929.

Baochuan Lu, Publications Chair
(2024), blu@sbuniv.edu, Division of
Computing & Mathematics, Southwest
Baptist University, Bolivar, MO 65613.

Ed Lindoo, Treasurer (2026),
elindoo@regis.edu, Anderson College of
Business and Computing, Regis
University, Denver, CO 80221.

Cathy Bareiss, Membership Secretary
(2025),
cathy.bareiss@betheluniversity.edu,
Department of Mathematical &
Engineering Sciences, Bethel University,
Mishawaka, IN 46545.

Judy Mullins, Central Plains
Representative (2026),
mullinsj@umkc.edu, University of
Missouri-Kansas City, Kansas City, MO
(retired).

Michael Flinn, Eastern Representative
(2026), mflinn@frostburg.edu,
Department of Computer Science &
Information Technologies, Frostburg
State University, Frostburg, MD 21532.

Representative (2025),
dnaugler@semo.edu, Brownsburg, IN
46112.

David Largent, Midwest
Representative(2026),
dllargent@bsu.edu, Department of
Computer Science, Ball State University,
Muncie, IN 47306.

Mark Bailey, Northeastern
Representative (2025),
mbailey@hamilton.edu, Computer
Science Department, Hamilton College,
Clinton, NY 13323.

Shereen Khoja, Northwestern
Representative(2024),
shereen@pacificu.edu, Computer
Science, Pacific University, Forest Grove,
OR 97116.

Mohamed Lotfy, Rocky Mountain
Representative (2025),
mohamedl@uvu.edu, Information
Systems & Technology Department,
College of Engineering & Technology,
Utah Valley University, Orem, UT
84058.

Tina Johnson, South Central
Representative (2024),
tina.johnson@mwsu.edu, Department of
Computer Science, Midwestern State
University, Wichita Falls, TX 76308.

Kevin Treu, Southeastern
Representative (2024),
kevin.treu@furman.edu, Department of
Computer Science, Furman University,
Greenville, SC 29613.

Michael Shindler, Southwestern
Representative (2026), mikes@uci.edu,
Computer Science Department, UC
Irvine, Irvine, CA 92697.

David R. Naugler, Midsouth

Serving the CCSC: These members are serving in positions as indicated:

Bin Peng, Associate Editor, bin.peng@park.edu, Department of Computing and Mathematical Sciences, Park University, Parkville, MO 64152.

Brian Hare, Associate Treasurer & UPE Liaison, hareb@umkc.edu, School of Computing & Engineering, University of Missouri-Kansas City, Kansas City, MO 64110.

George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of

Computer Science, Hood College, Frederick, MD 21701.

Megan Thomas, Membership System Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.

Karina Assiter, National Partners Chair, karinaassiter@landmark.edu, Landmark College, Putney, VT 05346.

Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Gold Level Partner

Rephactor

ACM2Y

ACM CCECC

Welcome to the 2024 CCSC South Central Conference

The 2024 South Central Steering Committee is very pleased to welcome everyone to our 35th annual conference hosted by Stephen F. Austin State University in Nacogdoches, Texas. Our conference chair and host, Anne Marie Eubanks, has provided infrastructure and support for the in-person delivery of our conference this year.

For our 2024 conference, we have nine papers, one tutorial, and several student and faculty posters scheduled for the program. This year, the Steering Committee chose 9 of 14 papers through a double-blind review process for a paper acceptance rate of 64%. Thirteen colleagues across the region and country served as professional reviewers, and we recognize the expertise and guidance they all so thoughtfully contributed to the selection of our 2024 conference program.

The Steering Committee continues to seek colleagues to host the conference in the future and to join our community of computer science educators to enrich our curricula and provide innovative pedagogy for our students. We invite and encourage our fellow members of the South Central region to attend our steering committee business meeting on Friday, April 5, 2024, after the conference reception and banquet. Fellow educators and colleagues are encouraged to join in our efforts to involve more of our community in the planning and execution of the conference in the future.

We extend a warm and delightful welcome to our presenters and attendees who continue to promote computer science education and camaraderie in our region. To all members of our 2024 Steering Committee, thank you again for your help organizing the conference and your gracious efforts in delivering our conference during such challenging times.

Anne Marie Eubanks
Stephen F. Austin State University
Conference Chair and Host

Bingyang Wei
Texas Christian University
Regional Editor Co-Chair

Mustafa Al Lail
Texas A&M International University
Regional Editor Co-Chair

2024 CCSC South Central Conference Steering Committee

Conference Chair

Anne Marie Eubanks Stephen F. Austin State University, TX

Past Conference Chair

Anne Marie Eubanks Stephen F. Austin State University, TX

Papers Chair

Bingyang Wei Texas Christian University, TX

Mustafa Al-Lail Texas A&M International University, TX

Reviewer Chair

Lasanthi Gamage Webster University, MO

Panels and Tutorials Chair

Jeffrey Zheng Stephen F. Austin State University, TX

Posters Chair

Shyam Karrah The University of Texas at Dallas, TX

Moderator Chair

Vipin Menon McNeese State University, LA

Publicity Chair

Eduardo Colmenares-Diaz Midwestern State University, TX

Nifty Assignments Chair

Michael Kart St. Edward's University, TX

At-Large Member

Vacant

Regional Board — 2024 CCSC South Central Region

National Board Representative

Tina Johnson Midwestern State University, TX

Registrar

Anne Marie Eubanks Stephen F. Austin State University, TX

Treasurer

Vacant

Regional Editor

Bingyang Wei Texas Christian University, TX

Mustafa Al-Lail Texas A&M International University, TX

Webmaster

Christian Servin El Paso Community College, TX

Reviewers — 2024 CCSC South Central Conference

Steve Cole Washington University in St. Louis, St. Louis, MO
Eduardo Colmenares-Diaz ... Midwestern State University, Wichita Falls, TX
Jason Dill Webster University, Webster Groves, MO
David Gurney Southeastern Louisiana University, Hammond, LA
Essa Imhmed Eastern New Mexico University, Portales, NM
Tina Johnson Midwestern State University, Wichita Falls, TX
Shyam Karrah The University of Texas at Dallas, Dallas, TX
Michael Kart St. Edward's University, Austin, TX
Tim McGuire Texas A&M University, College Station, TX
José Metrôlho
..... Instituto Politécnico de Castelo Branco, Castelo Branco, Portugal
Christian Servin El Paso Community College, El Paso, TX
Bilal Shebaro St. Edward's University, Austin, TX
Bill Siever Washington University in St. Louis, St. Louis, MO

The Evolution of IT in Higher Education and Preparing for the Future*

Opening Keynote

Michael Coffee
CIO, Stephen F. Austin State University

Bio

Michael Coffee has served in numerous capacities across IT in his thirty-year career spanning both private industry and higher education. During this time, he has witnessed significant changes in the IT industry and eagerly anticipates its future direction. Currently, Mike serves as the Chief Information Officer at Stephen F. Austin State University, leading the Information Technology Services division. Their mission is to act as trusted advisors and deliver robust IT services to all members of the university community. Mike holds a Bachelor of Science in Computer Science and a Master of Business Administration from Stephen F. Austin State University.



*Copyright is held by the author/owner.

A Case Study on Adopting Best Practices in Introductory Computer Science *

Jeremy Becnel
Department of Computer Science
Stephen F. Austin State University
Nacogdoches, TX 75962
becneljj@sfasu.edu

Abstract

This article discusses observations from a reimagining of the introductory computer science principles course at Stephen F. Austin State University. The course applies the best principles from a National Science Foundation study in conjunction with the College Board. Student attitudes are observed along with the student performance in the course and following courses.

1 Introduction

In 2011, the National Science Foundation (NSF) embarked on an initiative ¹ to broaden participation in computer science, targeting high schools. This effort culminated in the design of a new introductory computing course, Computer Science Principles (CSP). The primary aim of this course was to reach a broad audience of students. It was crafted to include rich computer science content and engaging pedagogy while maintaining the rigor and high standards of the Advanced Placement (AP) program ².

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

¹https://www.nsf.gov/news/news_summ.jsp?cntn_id=129882#

²https://www.nsf.gov/awardsearch/showAward?AWD_ID=1246919#

The Computer Science Principles ³ course was developed through a unique collaboration between the National Science Foundation, the College Board, and computer science educators at a number of universities. It was initially piloted nationwide and later became the newest College Board Advanced Placement offering during the academic year 2016-17 [3].

The project investigated the implementation and outcomes of Concurrent Enrollment (CE) programs ⁴. In addition to the AP course, an accompanying introductory course at the college level was developed employing best practices in computer science education [4]. This initiative represents a significant effort to integrate computer science education into the mainstream curriculum, thereby making it more accessible and appealing to a wider range of students and reaching underserved populations.

In this work, we take the strategies and lessons learned by this study and put them into practice for students served at Stephen F. Austin State University (SFASU) to ascertain if these practices can have a positive impact on student outcomes, attitudes, and perceptions.

1.1 Pilot Courses

As part of the NSF student, pilot courses at several universities were designed around these organizing principles: (1) Computing is a creative human activity that engenders innovation and promotes exploration. (2) Abstraction reduces information and detail to focus on concepts relevant to understanding and solving problems. (3) Data and information facilitate the creation of knowledge. (4) Algorithms are tools for developing and expressing solutions to computational problems. (5) Programming is a creative process that produces computational artifacts. (6) Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems. (7) Computing enables innovation in other fields including science, social science, humanities, arts, medicine, engineering business.

Additionally, the courses incorporated the following six computational thinking practices: (1) Analyzing the Effects of Computation (2) Creating Computational Artifacts (3) Using Abstractions and Models (4) Analyzing Problems and Artifacts (5) Communicating Processes and Results (6) Working Effectively In Teams.

Translating the broad concepts of the Seven Big Ideas and Six Computational Thinking Practices into a practical curriculum required an intermediate step of developing learning objectives. This groundwork paved the way for the creation of specific classroom materials and activities, tailored into curric-

³<https://new.nsf.gov/events/ap-computer-science-principles#>

⁴https://www.nsf.gov/awardsearch/showAward?AWD_ID=1837112#

ula by instructors at five diverse universities. The five pilot schools and their respective instructors included:

- University of North Carolina at Charlotte - Tiffany Barnes, teaching “The Beauty and Joy of Computing” using BYOB Scratch.
- University of California, Berkeley - Dan Garcia and co-instructor Brian Harvey, offering the same course as UNC Charlotte.
- Metropolitan State College, Denver - Jody Paul, teaching “Living In A Computing World” with Scratch.
- University of San Diego - Beth Simon, instructing “Fluency with Information Technology” using Alice.
- University of Washington, Seattle - Larry Snyder, teaching “Computer Science Principles” with Processing, in collaboration with high school teacher Susan Evans.

1.2 Extensions

Since the culmination of the program, several universities have modeled their introductory computer science course based on the lessons learned from the pilot studies. Some notable successes include Stony Brook University. The university previously offered a course, Introduction to Computational Arts ⁵, that introduces the fundamentals of programming, specifically focusing on combining arts and computing.

Another notable adaptation occurred at Bryn Mawr College. A National Science Foundation-sponsored survey found that students in an introductory computing course taught at Bryn Mawr College were twice as likely to take another computer science class compared to students in a class with a more traditional curriculum [10].

Possibly the greatest success occurred at Harvey Mudd College [1]. Harvey Mudd University saw an increase in the percentage of female majors from 10% to 40% after adapting their introductory course [6].

1.3 Study Summary

The introductory course in computer science at SFASU has traditionally followed the standard approach. That is, students are introduced to basic concepts such as branching and repetition in a lecture format and practice these concepts outside of class by constructing simple programs that read input and produce output either to a file or console.

⁵<https://www.my-mooc.com/en/mooc/compartprocessing/>

In conjunction with an NSF S-STEM proposal, the author piloted a course that made use of current best practices in computer education and incorporated aspects of the pilot course and the accompanying NSF/College Board study referenced previously. Concepts and practices were selected to best fit the students who typically attend the institution. Survey data and student tracking are examined to ascertain the success of the newly designed course.

2 Case Study

The pilot course was first facilitated in the Fall of 2021. The course was piloted to a group of eleven honors students. The following year, in the Fall of 2022, the course was run again in two regular sections consisting of 59 total students. The students were a mix of several different majors including computer science, nursing, mathematics, general studies, biology, engineering, chemistry, and others. The vast majority of students had no prior programming experience.

2.1 Reimagined Course Structure

Several key decisions were made regarding the course structure to ensure the course simultaneously included best practices, adhered to department requirements for content, and adequately prepared SFASU students for success in the following courses.⁶

The first key decision was to select the tools to use for the course. The courses that follow involve object-oriented programming and event-driven programming and use Java and C#, respectively. Thus the Processing (Java) programming environment/language was selected so students would learn the syntax and structures common to these statically typed programming languages. Processing is also easy to use, install, and has the added benefit of making creative/artistic tasks accessible to beginning programmers.

The content was the same as the traditional offerings of the course. However, the delivery has key differences. In place of a standard lecture approach, concepts were actively learned [7] using a “sandwich approach”. That is, students were briefly introduced to a concept by the instructor, worked on the concept with classmates, and then the last part of the class was devoted to discussion on the concept.

Assignments were frequent, typically two or three per week, and varied from worksheets, coding, and reading/writing. A notable alteration was the use of creative coding assignments, such as creating a logo or a simple game [5]. Main concepts typically had two related coding assignments (in place of standard assignments that read/write from the file/console). For each main concept

⁶Course materials are available from the author upon request.

(e.g. arrays, loops), the first assignment was a paired programming assignment where students could get comfortable with the assigned topic and the second was an individual assignment for the student to demonstrate mastery [11].

To provide assistance to students and serve as a role model/mentor, diverse upper-level and graduate students were selected to act as mentors. The addition of mentors assured students could receive help promptly whether in-person or through a course Discord server. The use of these mentors also helped to reduce gaps in prior experience [8].

The class also introduced students to “Big Ideas” in computing. This included security, how computers store information, computing for social good, algorithms, etc. Students were typically asked to read articles and participate in a short lecture by the instructor. Afterward, students were put in small groups and asked to make a 3-5 minute presentation on something that interested them about the topic. These activities helped to improve student anxiety about public speaking, fostered a collaborative environment, and improved student communication skills.

The course culminated in a pair programming project of the student’s choosing. The students presented their work during final exam week. The only other deviation from the traditional course offering was the use of short frequent quizzes in place of large monthly exams. In the following, we refer to the course outlined in this section as *reimagined* and the traditional lecture-based course as *traditional*.

2.2 Data Collection and Evaluation

In addition to examining course grades and feedback on course evaluations, we make use of two other means of evaluation: 1) Pre and Post Course Surveys were administered to students. In addition to demographic information, the survey asked students their opinions on the course, and their attitude toward computer science, and asked general questions about computer science in society. 2) Students who decided to continue their studies in the following course were tracked to determine their level of success.

3 Results

The grade distribution of the 59 students in the reimagined course can be found in Figure 1. Compared to 65 students who enrolled in a traditional section of the course, there were no statistically significant differences, other than in the number of F’s and W’s (drops). The traditional offering saw a larger percentage of F’s while the re-imagined course saw a larger number of drops (W). The instructor reached out to the 11 students that dropped and the

most common response was the course was more work than their other courses or more work than they expected.

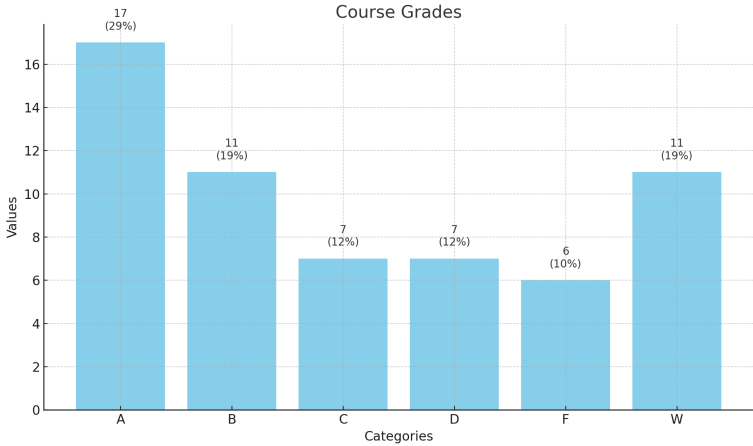


Figure 1: Course grade distribution (W is for students that dropped)

The courses that follow the introductory computer science principles course are Event Driven Programming and Object-Oriented Programming. To enroll in these courses students must earn a grade of C or better in the introductory course. There was no statistically significant difference in the percentage of students who earned a grade of C or better between the reimagined and traditional courses. While these initial results were somewhat disappointing, it is not uncommon for active learning techniques, such as flipped classrooms to show minimal impact on course grades [9]. However, active learning has been shown to lead to future success, improved attitudes, reduced anxiety, etc. [2]. We explore these in the next section.

3.1 Success in Future Courses

Of the 59 students who took the re-imagined introductory computer science course, 10 went on to take Object Oriented Programming and 15 went on to take Event Driven Programming. (Note: the students are not mutually exclusive.) The grade distribution of the students in these courses can be found in Table 1.

Even given the small sample size the number of students earning an A, the number of students passing (earning a D or better), and the number of students earning a C or better show a statistically significant improvement ($p = 0.05$) over historical departmental percentages.

Table 1: Grades in Following Course

Course/Grade	A	B	C	D	F	W (drop)
Event Driven	10	2	1	2	0	0
Object Oriented	7	2	0	1	0	0

3.2 Student Attitudes

Students in the reimagined course participated in pre and post-course surveys. ⁷ Students were asked on both surveys “In your opinion, what things someone who is skilled in computing can do. Please list 3 or more.” In the pre-course survey student responses primarily consisted of building websites, animation, visual design, and fixing computers. The post-course survey answers primarily consisted of course topics, such as working with arrays and images, working in groups, writing and understanding code, and making use of abstraction and generalization.

Students were also asked to respond to several Likert scale questions concerning the perception of their abilities regarding computer science. In the survey, we witness an increase from pre to post in the students’ perception of their abilities to productively work in a team, effectively perform research, solve logic problems, and work with interactive media.

The most significant improvement in student perception was found in the answers to the following questions 1) I can write successful computer programs. 2) I can effectively analyze the ethical, legal, and social implications of computing. The pre and post-survey results can be found in Table 2.

Table 2: Survey Results

Question	pre/post	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1)	pre	3%	20%	32%	30%	15%
1)	post	17%	53%	26%	2%	2%
2)	pre	17%	31%	41%	7%	4%
2)	post	30%	57%	11%	2%	0%

3.3 Concluding Remarks

While the pass rate and grade distribution did not demonstrate a significant difference between the reimagined course and the traditional course offering,

⁷Survey and results available from the author upon request.

there does seem to be some evidence that students who successfully complete the reimagined course perform better in subsequent courses. Hence this case study warrants a larger study on long-term success and retention.

Survey results demonstrate participation in the reimagined course results in positive changes in student attitudes and perceptions. Anecdotal evidence from course evaluations provides further evidence that students enjoy the course environment, the active learning methodology, and the creative assignments. Thus, the course adaptations in the reimagined course have the potential to lead to higher retention of computer science (and related) majors and recruitment of students from other disciplines, particularly from underserved groups. To verify the efficacy of the preliminary results in this case study a multi-year study should be undertaken to compare results to both historical data and a control group of students taking a traditional introduction computer science course.

References

- [1] Dodds Z. Alvarado C. and Libeskind-Hadas R. “Increasing Women’s Participation in Computing at Harvey Mudd College”. In: *ACM Inroads* 3 (2012), pp. 55–64.
- [2] Nanah Apkarian et al. “What really impacts the use of active learning in undergraduate STEM education? Results from a national survey of chemistry, mathematics, and physics instructors”. In: *PLOS ONE* 16 (Feb. 2021), e0247544. DOI: 10.1371/journal.pone.0247544.
- [3] Owen Astrachan and Rebecca Osborne. “Computer Science principles: portfolio-based assessment”. In: *ACM SIGCSE Bulletin* 44 (Oct. 2012), pp. 4–14. DOI: 10.1145/2398328.2398330.
- [4] Owen Astrachan and Rebecca Osborne. “Sorting in High School: The Good, The Bad, The Terrible”. In: Presented at the CSTA Conference in Omaha, NB, 2018.
- [5] Nathalia da Cruz Alves, Christiane Gresse von Wangenheim, and Lúcia Pacheco. “Assessing Product Creativity in Computing Education: A Systematic Mapping Study”. In: *Informatics in Education* 20 (Mar. 2021), pp. 19–45. DOI: 10.15388/infedu.2021.02.
- [6] Maria Klawe. *How Can We Encourage More Women to Study Computer Science?* 2015. URL: <https://www.newsweek.com/how-can-we-encourage-more-women-study-computer-science-341652> (visited on 01/06/2024).

- [7] Yin-Chan Liao and Marjorie Ringler. “Backward design: Integrating active learning into undergraduate computer science courses”. In: *Cogent Education* 10 (Apr. 2023). DOI: 10.1080/2331186X.2023.2204055.
- [8] Diba Mirza et al. “Undergraduate TA and Mentor Programs in Computer Science”. In: *SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Feb. 2019), pp. 1250–1250. DOI: 10.1145/3287324.3293744.
- [9] Elizabeth Setren et al. “Effects of Flipped Classroom Instruction: Evidence from a Randomized Trial”. In: *Education Finance and Policy* 16.3 (July 2021), pp. 363–387. ISSN: 1557-3060. DOI: 10.1162/edfp_a_00314. eprint: https://direct.mit.edu/edfp/article-pdf/16/3/363/1928207/edfp_a_00314.pdf. URL: https://doi.org/10.1162/edfp%5C_a%5C_00314.
- [10] Chris Stephenson et al. *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions*. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 9781450388320.
- [11] Jinbo Tan, Lei Wu, and Shanshan Ma. “Collaborative dialogue patterns of pair programming and their impact on programming self-efficacy and coding performance”. In: *British Journal of Educational Technology* (Dec. 2023). DOI: 10.1111/bjet.13412.

Fostering Code Quality Practices Among Undergraduate Novice Programmers*

*Essa Imhmed¹, Edgar Ceh-Varela¹,
Hashim Abu-Gellban², and Scott Kilgore¹*

*¹Mathematical Sciences Department
Eastern New Mexico University
Portales, NM 88130*

{essa.imhmed,eduardo.ceh,scott.kilgore}@enmu.edu

*²Computer Science Department
Grand Canyon University
Phoenix, AZ 85017*

hashim.abugellban@my.gcu.edu

Abstract

This paper reports on a study conducted to incorporate code quality into an introductory Java programming course. The discussion focuses on the strategies we used to teach coding standards necessary for writing high-quality code. We also present data and an analysis, investigating code quality issues identified through *CheckStyle* and *PMD* in students' code submissions. Our analysis revealed that 42% of code quality issues are related to code formatting and documentation, with an average of 363.19 issues per Thousand Lines of Code (*KLOC*) and 49.96 issues per *KLOC*, respectively. The analysis also revealed the presence of error-prone and other best practices issues. This analysis provides insight into the effectiveness of these teaching strategies.

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Coding standards are industry guidelines and best practices that become crucial to software development. These standards cover various aspects, including naming conventions, formatting, indentation, and comments/documentation. These aspects define how to write high-quality code improved for readability. Thus, adhering to coding standards can enhance team communication and reduce the implementation and maintenance costs and the risk of software bugs.

Similarly, code quality is of significant importance in academia. It helps course instructors understand their students' codes and, hence, efficiently review them for grading and timely feedback. Code quality also promotes effective collaboration among class project teams, as each student can comprehend the code written by their peers.

Despite that, code quality is often neglected in many university-level programming courses [3], exacerbated by several factors discussed in the literature. For instance, the findings from [12] indicate that most students do not consider coding standards a top priority and often fail to comply with them. Students justify this negligence with tight schedules and minimal grading emphasis on code quality. This issue is exacerbated by instructors who should enforce code quality in their courses. Instead, they tend to focus only on designing programming assignments that only assess students' problem-solving abilities rather than including assessments of their code quality [10, 11].

To address this issue, we integrated teaching strategies focused on coding standards into an introductory Java programming course. This aims to convey the importance of code quality early on among students learning to code. These strategies involve designing practical assignments to assess students' learning of coding standards and a grading rubric that enforces code quality in their code submissions. This paper reports on implementing these teaching strategies and presents data and an analysis, investigating code quality issues in students' programming assignment solutions. The results obtained from the study provide valuable insights into the effectiveness of these strategies.

The remainder of this paper is structured as follows. Section 2 presents research background. Section 3 summarizes the research methodology. Next, in Section 4, we discuss the result of this study. Section 5 concludes the paper.

2 Background

Recent studies have investigated code quality in an academic environment. Hofbauer et al. [7] report an analysis of source code submitted by novice students of a software engineering course. Their analysis revealed that students exhib-

ited an understanding of most of the software engineering concepts, but their code suffered readability and adherence to coding standards, specifically naming conventions. Findings from Li et al. [12] indicate that students recognize the importance of coding standards; however, there is a noticeable tendency among students to overlook compliance with these standards. Students often prioritize writing functional code over code that emphasizes readability [11]. Therefore, there is a need for teaching strategies addressing such discrepancies.

To address this issue, Li et al. explored several teaching strategies to enhance students' perception of code quality, including focusing on a small set of common coding standards suitable for novice programmers, providing students with a document that briefly outlined expectations on these standards, and assigning practical exercises on coding standards. Stegeman, Barendsen, and Smetsers [16] propose grading rubrics to enforce code quality in student code submissions. Such a rubric provides a systematic approach for grading and feedback, allowing students to greatly benefit from it [15]. Li et al. [12] also suggests adopting peer assessment approach among students to get peer feedback.

Furthermore, Chen et al. [4] propose using static analysis tools to help novice and expert programmers improve code quality. Oskouei et al. [13] compared three open-source static analysis tools for Java programs, namely: *PMD* [14], *FindBugs* [6], and *CheckStyle* [2]. Oskouei et al. found that each tool uncovers different kinds of bugs. *PMD* uncovers common programming flaws such as empty catch blocks and unused variables. *FindBugs* discovers bugs and potential runtime. *CheckStyle* detects code style violations.

Albluwi et al. [1] investigated the code quality of students in an introductory programming course at Princeton University. Utilizing *CheckStyle* and *PMD*, their results show common frequent errors among students related to formatting and documentation regardless of performance or degree major, with female students and in general students who work with peers producing fewer documentation and style errors. Edwards et al. [5] also utilized *CheckStyle* and *PMD* to investigate the code quality of over 500 thousand programs submitted across four computer science courses at Virginia Tech. Their analysis revealed the presence of 10 million coding standards violations in students' codes, of which formatting and *JavaDoc* documentation errors are the most frequent and are consistent between computer science majors and non-majors across different experience levels.

3 Methodology

Figure 1 depicts an overview of the study process. Due to space limits, we provide a very abbreviated discussion of the study process below.

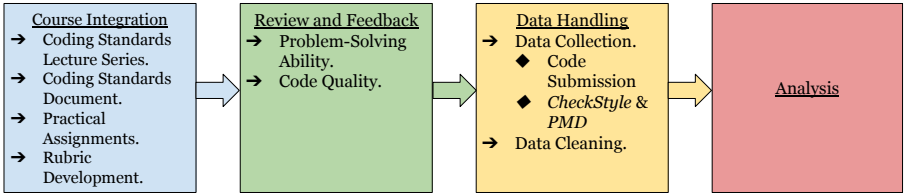


Figure 1: Overview over the study process.

3.1 Course Integration

To help students develop the habit of writing high-quality code, we incorporated coding standards into an introductory programming course at Eastern New Mexico University. The course aims to introduce students to problem-solving, algorithmic-thinking, and software development using the Java programming language, covering basic programming concepts such as data types, conditions, loops, methods, arrays, and File I/O. Therefore, we delivered a series of lectures featuring coding standards with practical code examples. Similar to [12], we introduced students to a small set of coding standards suitable for novice programmers yet typical for completing programs with medium complexity. We also provided students an access to a coding standards document and instructed them to comply with it when writing code.

Listing 1: An example of coding standards practical assignment

```

1  import java.util.Scanner;
2  //This program reads, scales, and reverses a sequence of numbers.
3  public class Main
4  {public static void main(String[] args )
5  {
6      double[] a=read_inputs(5);multiply(a, 10);printreversed(a);
7  }
8  // Reads a sequence of floating-point numbers.
9  // n is the number of inputs to read
10 public static double[] read_inputs(int n)
11 {
12     System.out.println("Enter_"+n+"_numbers:_");
13     Scanner in=new Scanner(System.in); double[] inputs=new
14         double[n]; for(int
15         i=0;i<inputs.length;i++){inputs[i]=in.nextDouble();} return
16         inputs;
17 }
18 //Multiplies all elements of an array by a factor.
19 public static void multiply(double[] values,double factor)
20 {
21     for(int i=0;i<values.length;i++){values[i]=values[i]*factor;}
22 }
23 //Prints an array in reverse order.
24 public static void printreversed(double[] v)
25 {// Traverse the array in reverse order, starting with the last
26     element
27     for(int i=v.length-1;i>=0;i--)
28     {System.out.print(v[i]+"_");}
29     System.out.println();
30 }
31 }
  
```

Furthermore, we assigned the students a couple of practical assignments focused on coding standards. Listing 1 presents one of these assignments. As listed, the program functions correctly but contains several violations of coding standards, such as line length issues, white space problems, and lack of JavaDoc comments. The student’s task is to identify these violations and resubmit the program after addressing them. Once submitted, we review students’ solutions with particular attention to their compliance with coding standards.

Table 1: Coding standards criterion from the grading rubric

Criteria	Ratings
Coding Standards	10 pts Excellent (100%) Clearly and effectively documented, including the JavaDoc tags for the author name, date, and assignment title; includes descriptive names of all program variables and functions; specific purpose noted for each function, control structure, input requirements, and output results; excellent use of white space; creatively organized work.
	6 pts Satisfactory (60%) , but needs Improvement It includes JavaDoc tags for the author name, date, and assignment title; basic documentation has been completed, including descriptive names of all program variables and functions; purpose is noted for each function; white space makes the program fairly easy to read; organized work.
	4 pts Unsatisfactory ($\leq 40\%$) No author name, date, or assignment title is included; very limited or no documentation is included; documentation and naming do not help the reader understand the code; poor use of white space (indentation, blank lines); disorganized and messy.

Similar to [16], we also developed a grading rubric that assesses students’ problem-solving ability and code quality. Table 1 shows the coding standards criterion from the rubric. As listed in the table, we focused on a small set of coding standards related to documentation/comments, size violations (e.g., Line length), naming conventions, white space, and code block violations. We used the rubric to enforce code quality in all students’ solutions to programming assignments.

3.2 Static Analysis Tool Selection

Static analysis is a process in which a program’s source code is examined for potential bugs or violations without executing it. For this study, we selected

CheckStyle and *PMD* to identify coding standards violations in students’ solutions of programming assignments. *CheckStyle* and *PMD* are highly configurable static analysis tools, allowing programmers to extend or choose the types of violations these tools should detect. Following the approach in [1], we used *CheckStyle* to check source code for coding conventions and style and *PMD* to identify non-simplified expressions and code issues that may lead to potential software bugs.

3.3 Data Collection

We collected 507 Java programming assignment solutions submitted by students over a 3-semester period from Fall 2022 through Fall 2023, totaling 27,963 lines of code. We performed static analysis on the code submissions using *CheckStyle* and *PMD*. We configured *CheckStyle* and *PMD* to check for violations of the coding standards set in the grading rubric. Subsequently, we obtained reports detailing 29,136 instances of coding standards violations. Each instance includes a file name and a code line number indicating where the violation has occurred. It also describes the violation and its category. Finally, we conducted data cleaning on the reports to generate a refined dataset containing only the description and category for each violation. Following this, we analyzed the new dataset.

4 Results and Analysis

This section presents a quantitative analysis of code quality issues identified in students’ code submissions, focusing particularly on code style, best practices, and error-prone violations.

Table 2: Error category

Category	Error
Formatting	White space around; line has trailing spaces; line length; whites pace after; padding of parentheses; method <i>ParamPad</i> ; no white space after; no white space before; new line at end of file; file tab character.
Documentation	Missing <i>Javaoc</i> method; invalid <i>JavaDoc</i> position; <i>JavaDoc</i> style; <i>JavaDoc</i> method; <i>JavaDoc</i> variable; <i>JavaDoc</i> type.
Naming Conventions	Local variable name; member name; local final variable name; method name; parameter name; static variable name.
Readability	Need braces; avoid nested blocks; empty catch block.

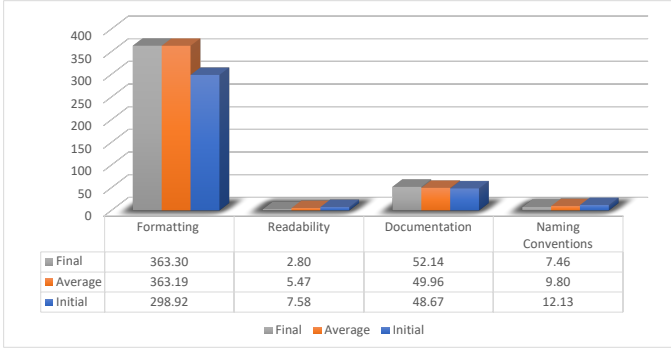


Figure 2: Error rates (in KLOC) on initial, average and then final submissions.

4.1 Code Style

Table 2 shows the error categorisation. Figure 2 presents a comparison of the average error category rate per Thousand Lines of Code (*KLOC*) across students' code submissions, compared to error frequencies in the initial and final semester submissions. The figure shows that Formatting errors are most common errors, with an average of 363.19 per *KLOC*, followed by Documentation errors at 49.96 per *KLOC*. Together, Formatting and Documentation errors account for up to 42% of the total errors in the student's codes. For a detailed description of the those category errors, see [2].

Looking at the final submissions of these two categories, it is clear that there is an overall increase in the error rate per *KLOC* compared to the initial submissions, with up to a 19.44% percentage increase in Formatting errors and a 6.88% percentage increase in documentation errors. This could result from having large programming assignments with more complex structures while progressing toward the end of the semester. However, it also suggests that, on average, students do not show improvement in these areas compared to other aspects, such as naming conventions.

Since both Formatting and Documentation categories combine several code style errors, further investigation is needed to identify which errors occur most frequently. Figure 3 compares the average frequency of Formatting errors across all code submissions. It is clear that the error *WhiteSpaceAround* presents as the most frequent Formatting errors at 158.28 per *KLOC*, followed by *LineHasTrailingSpace* at 72.56 per *KLOC*, and *LineLength* at 64.12 per *KLOC*, compared to the other Formatting errors across all code submissions. While these top three errors are considered harmless to code functionality, they impact code readability and collaboration among team members, making

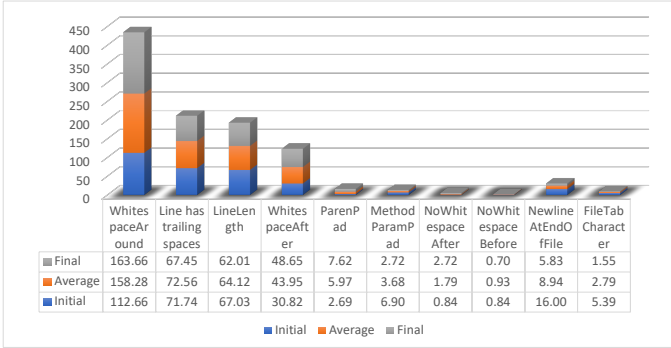


Figure 3: Formatting error rates (in KLOC) on initial, average and then final submissions.

it challenging for reviewers to comprehend modifications during code reviews and maintenance.

Figure 4 compares the average frequency of documentation errors across all code submissions. Despite students, on average, showing improvement in *MissingJavaDocMethod* and *InvalidJavaDocPosition* errors, compared to the initial submissions of the same category, they are still the most frequently occurring error at 19.60 per *KLOC* and at 9.66 per *KLOC*, respectively. *MissingJavaDocMethod* error indicates the absence of *JavaDoc* comments for a method. *InvalidJavaDocPosition* error indicates that a *JavaDoc* comment is placed at an incorrect position. *JavaDocStyle* error, representing a violation in the format of *JavaDoc* comments, is the third most frequent at 9.58 per *KLOC*. Our analysis revealed that some students do not adhere to *JavaDoc* syntax when documenting methods and classes. For instance, these students document the method using comment blocks instead of the corresponding *JavaDoc* tags. Also, some of them place *JavaDoc*'s comments describing the class inside the class structure instead of before the class declaration.

4.2 Error-Prone and Best Practice

Error-prone and best practice violations [14] are considered code issues that increase the risk of potential software bugs. Although not enforced in our rubric, our analysis reveals the presence of these errors in students' code submissions, highlighting the necessity to include these code quality aspects in our course. The topmost best practice violation identified is *SystemPrintln* error with 22.11 per *KLOC*, followed by *UnusedAssignment* at 1.88 per *KLOC*. It is recommended to use a logger instead of `System.out/err.println`, as the latter is

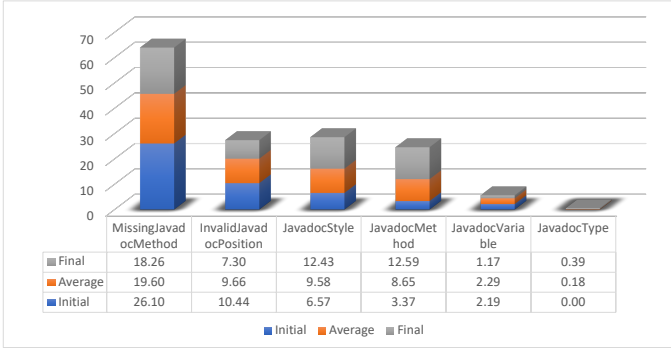


Figure 4: Documentation error rates (in KLOC) on initial, average and then final submissions.

usually intended for debugging purposes and can remain in the codebase even in production code [14].

Our analysis also revealed that *CloseResource* is the most frequent error-prone violation presented in students’ code submissions at 4.33 per *KLOC*. We noticed that several code submissions utilizing the method *Scanner* for reading from the console or files do not include statements that properly release them once the I/O operation is complete. This type of violation can lead to resource leaks, such as file locks or unexpected behavior causing the exhaustion of system resources [14].

4.3 Instructor Reflection

Given the average error rate in the code quality categories—namely, naming conventions, readability, and documentation—the integrated teaching strategies on coding standards have fostered code quality practices among students. However, our analysis also revealed that, on average, students did not show improvement over the semester in code formatting—mainly white space errors—and show slight improvement in code documentation, which could be a result from using integrated development environments (IDEs) with auto-completion features.

Our students used *Visual Studio Code (VSC)* to write their programs. We observed that the majority of them relied on the auto-completion features for indentation and code completion, including the construction of *JavaDoc* comments. Notably, *JavaDoc* comments for methods, in particular, can be correctly constructed only when the methods are entirely coded. This could be the reason why *WhiteSpaceAround* and *JavaDocStyle* errors are most fre-

quently presented in all submissions. Furthermore, the default settings in *VSC* do not enforce specific line length. This lack of enforcement may explain why *LineLength* error is among the most frequent errors observed in all submissions.

Another noteworthy observation is that some students often neglect to adhere to coding standards when these standards are not enforced by the grading rubric, such as during quizzes, exams, and lab sessions. Even when coding standards are enforced, a few students tend to refrain from complying with them, as the portion of the assignment grade assigned is relatively small, suggesting that the effort may not be worthwhile. Therefore, further research is needed to

1. Explore other teaching strategies, such as having students perform peer reviews on their code, to enhance students' proficiency in the formatting error category,
2. Integrate error-prone and best practices into our course,
3. Investigate in what context such IDEs impact the learning of novice programmers in this area.

5 Conclusion and Future Work

In this study, we integrated several teaching strategies on coding standards into an introductory programming course, aiming to enhance students' awareness and perceptions of code quality. We also presented data and analysis, investigating code quality issues among novice students and providing insight into the effectiveness of those strategies. Although the analysis revealed that these strategies helped students to improve in most of the quality categories, such as naming conventions and readability, we found that students still perform poorly in code formatting and documentation, particularly white space and *JavaDoc* comments for methods. In addition, students often neglect to comply with coding standards when they are optional and not enforced by the grading rubric. These findings align with those in [5, 1, 12]

The next step of this research involves automating the process of code quality assessment. We also intend to investigate students' code quality issues with the C/C++ programming languages. Thus, we plan to implement a scheme within the LLVM framework [8, 9] to detect code clones and best practice issues such as redundant operations and unused assignments in student codes during the compilation stage.

6 Acknowledgement

This research is funded by Eastern New Mexico University (ENMU) through the Faculty Research and Instructional Development Grant program. We would also like to thank Brian Pasko, of ENMU, for his contribution to review & edit the manuscript.

References

- [1] Ibrahim Albluwi and Joseph Salter. “Using static analysis tools for analyzing student behavior in an introductory programming course”. In: *Jordanian Journal of Computers and Information Technology (JJCIT)* 6.3 (2020), pp. 215–233.
- [2] *Checkstyle*. <https://checkstyle.sourceforge.io>. Accessed: 2024-1-7.
- [3] Hsi-Min Chen, Wei-Han Chen, and Chi-Chen Lee. “An Automated Assessment System for Analysis of Coding Convention Violations in Java Programming Assignments”. In: *J. Inf. Sci. Eng.* 34 (2018), pp. 1203–1221. URL: <https://api.semanticscholar.org/CorpusID:52165154>.
- [4] Hsi-Min Chen, Wei-Han Chen, and Chi-Chen Lee. “An Automated Assessment System for Analysis of Coding Convention Violations in Java Programming Assignments.” In: *J. Inf. Sci. Eng.* 34.5 (2018), pp. 1203–1221.
- [5] Stephen H. Edwards, Nischel Kandru, and Mukund B.M. Rajagopal. “Investigating Static Analysis Errors in Student Java Programs”. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 65–73. ISBN: 9781450349680. DOI: 10.1145/3105726.3106182. URL: <https://doi.org/10.1145/3105726.3106182>.
- [6] *FindBugs - Find Bugs in Java Programs*. <https://findbugs.sourceforge.net>. Accessed: 2024-1-7.
- [7] Markus Hofbauer et al. “Teaching software engineering as programming over time”. In: *Proceedings of the 4th International Workshop on Software Engineering Education for the Next Generation*. 2022, pp. 51–58.
- [8] Essa Imhmed, Jonathan Cook, and Abdel-Hameed Badawy. “Evaluation of a Novel Scratchpad Memory through Compiler Supported Simulation”. In: *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. 2022, pp. 1–7. DOI: 10.1109/HPEC55821.2022.9926335.

- [9] Essa Abubaker Imhmed. “Understanding Performance of a Novel Local Memory Store Design through Compiler-Driven Simulation”. PhD thesis. New Mexico State University, 2022.
- [10] Oscar Karnalim et al. “Promoting code quality via automated feedback on student submissions”. In: *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2021, pp. 1–5.
- [11] Oscar Karnalim, William Chivers, et al. “Work-In-Progress: Code Quality Issues of Computing Undergraduates”. In: *2022 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2022, pp. 1734–1736.
- [12] Xiaosong Li and Christine Prasad. “Effectively teaching coding standards in programming”. In: *Proceedings of the 6th conference on Information technology education*. 2005, pp. 239–244.
- [13] Elmira Hassani Oskouei and Oya Kalipsiz. “Comparing Bug Finding Tools for Java Open Source Software”. In: (2018).
- [14] *PMD*. <https://pmd.github.io>. Accessed: 2024-1-7.
- [15] D Royce Sadler. “Formative assessment and the design of instructional systems”. In: *Instructional science* 18 (1989), pp. 119–144.
- [16] Martijn Stegeman, Erik Barendsen, and Sjaak Smetsers. “Designing a rubric for feedback on code quality in programming courses”. In: *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 2016, pp. 160–164.

A Mobile App Leveraging NLP Techniques for Sci-Fi Book Recommendations*

Edgar Ceh-Varela, Essa Imhmed and Drey Smith
Department of Mathematical Sciences
Eastern New Mexico University
Portales, NM, USA
{eduardo.ceh, essa.imhmed, drey.smith}@enmu.edu

Abstract

This paper introduces the development of a mobile application employing Natural Language Processing (NLP) techniques to provide content-based recommendations for Sci-Fi books. The mobile application integrates two distinct NLP techniques: Doc2Vec for rapid keyword searches and RoBERTa to enhance the understanding of book themes and ideas. This combination enables the recommender system to offer personalized book recommendations tailored to individual user interests, enhancing the reading experience.

1 Introduction

In today's world of mobile apps, finding personalized book recommendations is a common need. Recommender systems help by suggesting books or movies based on users' interests [4]. However, there remains a considerable gap in the availability of mobile applications tailored to recommend science fiction (Sci-Fi) books utilizing advanced natural language processing (NLP) techniques.

This research addresses this gap by introducing a new mobile app to recommend Sci-Fi books and novels. Leveraging NLP techniques, including Doc2Vec [7]

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

and RoBERTa [9], our app seeks to understand the unique content of Sci-Fi books more effectively. Our recommendations are based on the book’s content, presenting a novel contribution in personalized literary suggestions.

By examining the app’s construction, we aim to contribute valuable insights to the broader discourse on content-based recommender systems [12], presenting a different perspective on personalized book recommendations for Sci-Fi enthusiasts.

This paper is organized as follows: Section 2 presents the literature related to our research. Section 3 details the proposed method. The results are presented in Section 4 and our conclusions and future work in Section 5.

2 Background

Recommender systems (RS) represent a crucial area of research dedicated to suggesting relevant items by modeling user preferences and interests. In the context of modern mobile applications, RS has become ubiquitous, providing recommendations for multimedia content and products [8]. Different recommendation techniques, such as collaborative filtering, content-based filtering, and hybrid approaches, have been explored [12].

Mobile recommender systems face distinctive challenges compared to traditional platforms, given their constrained resources and the demand for personalized recommendations. Recent research has delved into mobile recommender systems across diverse domains, including tourism, news, and books [5]. Particularly in the realm of books, content-based recommendation has shown promise by leveraging natural language processing (NLP) to analyze textual descriptions and user reviews [1].

Document embedding techniques, such as Doc2Vec, have been applied to capture semantics and map documents to vectors for similarity analysis [7]. Concurrently, deep contextualized language models like BERT [3] have gained popularity in NLP, demonstrating high performance across various semantic tasks. RoBERTa, an improvement upon BERT, was trained on significantly more data for an extended period, leading to enhanced masked language modeling objectives and improved downstream task performance [9].

Despite the success of these NLP methods independently, limited efforts have explored their combined application in a mobile app for content-based Sci-Fi book recommendations. This paper proposes integrating Doc2Vec and RoBERTa to enable content-based filtering for science fiction novels. By capturing different semantics, these techniques aim to enhance recommendation accuracy. The development of this specialized mobile recommender system seeks to improve accessibility to new books through personalized, fine-tuned suggestions based on writing content.

3 Methodology

Figure 1 presents the components comprising the recommender system framework for our mobile application. Subsequent sections will provide detailed descriptions of each of these components.

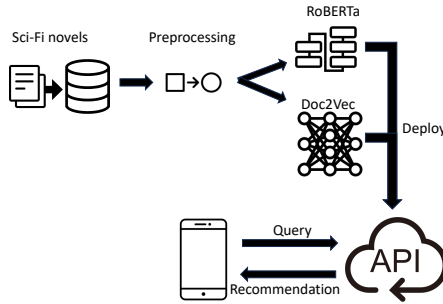


Figure 1: Framework for the integration of the mobile app and the recommender system

3.1 Dataset

For this project, the dataset comprises information and text from books and novels scraped from Project Gutenberg¹. We restricted the dataset to documents falling under the genre of Science Fiction and written in English. We collected the text of 1591 books and their accompanying information, including title and author. Since Project Gutenberg does not provide the year of publication, we manually added this information for each book in our dataset.

3.2 Preprocessing

Effective text preprocessing is a foundational task within any NLP system [2]. We removed text that Project Gutenberg inserted by default, transformed text to lowercase, stemmed and lemmatized words, and removed special characters and stop words. Finally, we concatenated each document’s title, author, and text as an additional feature of our dataset.

3.3 Text Embeddings

Text embeddings capture and represent semantic relationships and contextual information within textual data [6]. The embeddings are numerical encodings

¹<https://www.gutenberg.org/ebooks/bookshelf/68>

of the meanings of words, phrases, sentences, or even entire documents in a way that reflects the relationships between different words and concepts. These embeddings can then be applied to various natural language processing tasks, such as text classification, sentiment analysis, machine translation, and information retrieval [11].

For our application, we strategically integrated two powerful NLP approaches, Doc2Vec and RoBERTa, to enhance document similarity retrieval based on distinct user requirements.

Doc2Vec processes text, creating embeddings that encapsulate the semantic essence of a document or sentence. It is well-suited for short texts due to its context aggregation and fixed-length embeddings [10]. Using Doc2Vec, a neural network-based approach, we efficiently identify documents similar to user-provided input. When a user submits a query, the system can retrieve documents that closely match the user’s input by measuring the similarity between these embeddings. We implemented Doc2Vec using the Gensim² library in Python, configured with an embedding size of 100 and a window size of 4. The model was trained for 30 epochs to prevent overfitting.

RoBERTa’s transformer architecture and attention mechanisms are particularly effective for capturing intricate relationships in longer texts [9]. We utilize RoBERTa to find documents in our corpus that are similar to a given document selected by the user. The transformer architecture excels in capturing intricate contextual relationships within the text. RoBERTa enables us to grasp subtle semantic similarities between documents by leveraging pre-trained embeddings and fine-tuning them with our specific dataset. This approach proves invaluable when the goal is to identify documents closely related to a particular reference document. We employed the pre-trained RoBERTa model from Hugging Face³, generating embeddings of 768 dimensions.

3.4 Finding Similar Documents

We created two vector databases using the embeddings generated through Doc2Vec and RoBERTa. This strategic approach allows us to leverage the unique strengths of each model for two distinct input scenarios.

In the first scenario, for user-provided search strings, the system seamlessly transforms the input into its embedding representation using the Doc2Vec model. By harnessing the capabilities of Doc2Vec, the semantic essence of the search string is captured, enabling effective matching against the embeddings in the Doc2Vec vector database. This process facilitates identifying and retrieving documents closely aligned with the user’s textual query.

²https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html

³<https://huggingface.co/roberta-base>

In the second scenario, when users input a previous recommendation to find similar documents, the system employs the RoBERTa vector database to identify the most similar documents to the input document ID. RoBERTa excels in handling longer and more complex texts, allowing it to comprehensively capture subtle semantic relationships within the recommendation.

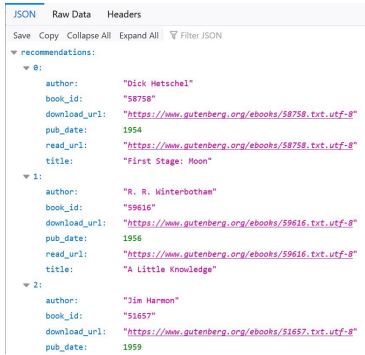
We use cosine similarity [13] to find the most similar embeddings. This measure is computed by evaluating the cosine of the angle between two vectors (i.e., document embeddings generated by Doc2Vec and RoBERTa), representing the degree of alignment. The cosine similarity is computed as,

$$\cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2} \tag{1}$$

where A and B are the embedding vectors.

The computed cosine similarity scores are the basis for our similarity-based recommendation algorithm. Higher cosine similarity scores indicate greater similarity between documents, influencing the recommendation model to prioritize content closely aligned with the input document or user query.

3.5 API and Mobile App Development



```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
▼ recommendations:
  ▼ 0:
    author: "Dick Hetschel"
    book_id: "58758"
    download_url: "https://www.gutenberg.org/ebooks/58758.txt.utf-8"
    pub_date: 1954
    read_url: "https://www.gutenberg.org/ebooks/58758.txt.utf-8"
    title: "First Stage: Hoon"
  ▼ 1:
    author: "R. R. Wintorboham"
    book_id: "59616"
    download_url: "https://www.gutenberg.org/ebooks/59616.txt.utf-8"
    pub_date: 1955
    read_url: "https://www.gutenberg.org/ebooks/59616.txt.utf-8"
    title: "A Little Knowledge"
  ▼ 2:
    author: "Jie Harmon"
    book_id: "51657"
    download_url: "https://www.gutenberg.org/ebooks/51657.txt.utf-8"
    pub_date: 1959
```

Figure 2: Example of JSON file returned by the API.

We developed a robust API that connects our recommendation system with the mobile application. Utilizing the Python Flask framework⁴, we hosted the API on the PythonAnywhere⁵ web platform. This framework enables us to make the recommender system accessible from anywhere on the internet, providing a user-friendly mobile application experience. We work with JSON files,

⁴<https://flask.palletsprojects.com/en/3.0.x/>

⁵<https://www.pythonanywhere.com>

a structured and efficient method for handling information, ensuring smooth communication between the API and the mobile application. This choice aligns with current data-sharing standards, guaranteeing compatibility and seamless integration with various applications and platforms.

Figure 2 shows the JSON file received as a response from the API when searching for similar novels to “Marty the Martian” by Arnold Marmor⁶. The information returned includes fields such as the author, book ID, title, publication date, and URLs for downloading the text from the Project Gutenberg website. Our API returns the top 10 most similar documents for our two user input scenarios (see Section 3.4).

To interact with the API, we developed an Android mobile application. This application allows users to receive book recommendations, view book information, add books to their personal library, and read them online and offline. The app is compatible with a minimum Android version of 7 (API 24) or higher.

4 Results

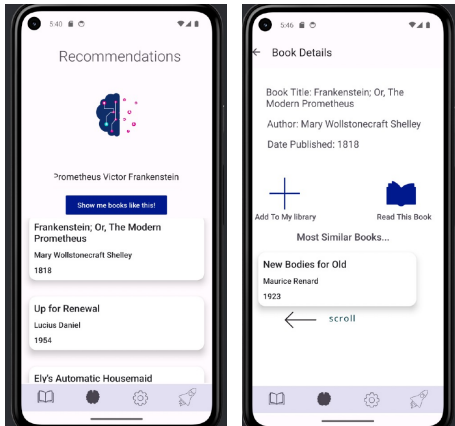


Figure 3: Mobile application for Sci-Fi novels recommendations

Figure 3 illustrates two interactions of a user with our mobile application. On the left, the user entered the string “*Prometheus Victor Frankenstein,*” into the mobile app. The input is then transformed into an embedding representation using Doc2Vec. The recommender system uses this embedding to find the top similar embeddings representing the documents in our dataset, which

⁶<https://www.gutenberg.org/ebooks/66389>

we have stored in a vector database. Then, the app presented a list of books and novels related to the user input. The figure shows that the first result was actually “*Frankenstein; Or, The Modern Prometheus*.”⁷ This outcome demonstrates that by using the Doc2Vec model, our recommender system effectively processes user inputs and identifies books with similar content.

On the right, when the user selects a recommended book or novel, the system provides a list of the most similar documents. The system has a vector database containing the RoBERTa embeddings for each document. The recommender system gets the precomputed embedding for the selected document and finds the top similar documents within the vector database. The figure indicates that the most similar document in our corpus to “*Frankenstein; Or, The Modern Prometheus*,” is “*New Bodies for Old*.”⁸ This result is a good recommendation, as both novels primarily focus on the intersection of science, technology, and human mortality. Therefore, we observe that using the RoBERTa model is valuable when the task is to find similar documents considering their entire text.

5 Conclusions and Future Work

By combining the strengths of Doc2Vec and RoBERTa, our recommender system offers diverse document similarity solutions for our mobile app. Whether users are searching for documents related to their input or seeking documents similar to a given reference, our dual-model approach ensures a robust and comprehensive document retrieval experience. This innovative combination enables us to accommodate to diverse user requirements, providing a flexible and efficient solution for document similarity retrieval across various scenarios.

In our future research, we plan to assess the effectiveness of our system’s recommendations by seeking feedback from Sci-Fi book enthusiasts. This evaluation will focus on understanding whether the suggested recommendations prove helpful for readers in discovering new and engaging books within the Sci-Fi genre.

References

- [1] Melania Berbatova. “Overview on NLP techniques for content-based recommender systems for books”. In: *Proceedings of the Student Research Workshop Associated with RANLP 2019*. 2019, pp. 55–61.

⁷<https://www.gutenberg.org/ebooks/84>

⁸<https://www.gutenberg.org/ebooks/59647>

- [2] Christine P Chai. “Comparison of text preprocessing methods”. In: *Natural Language Engineering* 29.3 (2023), pp. 509–553.
- [3] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).
- [4] Folasade Olubusola Isinkaye, Yetunde O Folaajimi, and Bolande Ade-fowoke Ojokoh. “Recommendation systems: Principles, methods and evaluation”. In: *Egyptian informatics journal* 16.3 (2015), pp. 261–273.
- [5] Dietmar Jannach et al. “A survey on conversational recommender systems”. In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–36.
- [6] Tom Kenter and Maarten De Rijke. “Short text similarity with word embeddings”. In: *Proceedings of the 24th ACM international on conference on information and knowledge management*. 2015, pp. 1411–1420.
- [7] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.
- [8] Jiafeng Li et al. “Personalized mobile video recommendation based on user preference modeling by deep features and social tags”. In: *Applied Sciences* 9.18 (2019), p. 3858.
- [9] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).
- [10] Natalia Maslova and Vsevolod Potapov. “Neural network doc2vec in automated sentiment analysis for short informal texts”. In: *Speech and Computer: 19th International Conference, SPECOM 2017, Hatfield, UK, September 12-16, 2017, Proceedings 19*. Springer, 2017, pp. 546–554.
- [11] John X Morris et al. “Text embeddings reveal (almost) as much as text”. In: *arXiv preprint arXiv:2310.06816* (2023).
- [12] Michael J Pazzani and Daniel Billsus. “Content-based recommendation systems”. In: *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 325–341.
- [13] Ramni Harbir Singh et al. “Movie recommendation system using cosine similarity and KNN”. In: *International Journal of Engineering and Advanced Technology* 9.5 (2020), pp. 556–559.

Teaching Cross-Platform Mobile Development and Cultivating Self-Directed Learners – A Six-Week Summer Online Course Experience*

Liqiang Zhang
Computer and Information Sciences
Indiana University South Bend
South Bend, IN 46615
liqzhang@iu.edu

Abstract

The rising popularity of cross-platform mobile programming in both industry and classroom necessitates effective teaching methods. This paper outlines our experience in instructing mobile programming through Google’s Flutter framework in a six-week, fully online summer course. A significant challenge faced was striking a balance between the breadth and depth of topics within the constraints of the short summer duration, while ensuring students receive ample hands-on training for an optimal learning experience. We address this challenge by adopting a dual approach: thoughtful content arrangement and strategic pedagogical methods. Specifically, we successfully deliver content by blending app-driven and topic-driven lectures within weekly modules, supplemented by a mix of required and optional reading assignments. We foster students’ self-directed learning (SDL) abilities through carefully crafted hands-on assignments and an open-ended final project.

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Recent years we have witnessed increased interest in learning mobile app development from students. However, teaching mobile app development is never an easy task [8]. Compared to general-purpose programming taught in a typical CS1 or CS2 course, mobile app development technologies, frameworks, and best practices are often evolving at a much faster pace. While staying current with the latest updates and trends is crucial for mobile app developers, this poses a special challenge to the course designer and instructor. Moreover, Mobile app development often involves targeting specific platforms such as iOS and Android. Each platform has its own set of development tools, languages (Swift for iOS, Kotlin/Java for Android), and design guidelines. A single course covering both Android and iOS programming might not always be a viable option due to the extensive list of topics to cover on each side.

In response to the launch of our Online Informatics program, we have recently undergone a comprehensive redesign of our Mobile App Development course. This initiative not only transitioned the course to a fully online format but also involved a shift in content—from Android programming to cross-platform mobile programming, utilizing Google’s Flutter framework [5]. Aligned with our commitment to nurturing self-directed learning (SDL) [2] abilities, which we consider essential in computing education to prepare students for the dynamic and rapidly evolving nature of the field, we have integrated elements into the course design that promote SDL. This paper aims to share our experience of the course design and lessons learned during the development and deployment processes over the past three summers.

2 Background

Cross-platform mobile development offers several advantages over native platform-specific development for Android or iOS, such as development efficiency, cost savings, and broader reach. This has made it an appealing choice for many developers and businesses. It also starts to gain popularity in the classroom as evidenced in some recent work [7, 8].

Self-directed learning (SDL), according to the definition given by Knowles in [6], is a process in which individuals take the initiative, with or without the help of others, in diagnosing their learning needs, formulating learning goals, identifying human and material resources for learning, choosing and implementing appropriate learning strategies, and evaluating learning outcomes. It is worthwhile noting that, to Knowles, SDL can be solely an individual effort, or it can incorporate the guidance of a mentor or the help of partners. Collier made the claim in [2] that SDL is a transformative learning pathway open to

all and that the skills of SDL are learnable and can be developed over time.

We are not the first to apply SDL in computing education. Examples of effort on fostering SDL in CS courses include [1, 9, 10]. The rapid evolution of mobile app development and wide range of topics, coupled with the diverse interests of our students and the constraints of time, reinforces our belief that the course presents an exceptional opportunity to cultivate students' SDL abilities.

3 Course Design and Development

Since the emergence of the first cross-platform framework around 2009, we have observed a significant proliferation and maturation of such frameworks. According to Statista [3], Flutter, React Native, Apache Cordova, Ionic, Xamarin, and Unity are among the most popular frameworks based on the data from 2019 to 2022. While each framework has its pros and cons, Flutter stands out in our consideration due to several compelling reasons including popularity, performance, hot reload feature, and strong community support.

In the rest of this section, we present our course design and development from several aspects, including topic selection, content delivery methods, the deliberate choice to forgo a textbook in favor of alternative resources, the structure of the course schedule and weekly modules, and the assessment of student learning. Throughout this exploration, we demonstrate the intentional integration of SDL principles into the fabric of the course design.

3.1 Topic Selection and Coverage

We compiled a list of topics pertinent to mobile programming with Flutter, which was subsequently prioritized based on the overarching learning objectives. The coverage plan is outlined in Figure 1. Notably, different levels of coverage are assigned to each topic: L1 signifies introductory coverage, L2 indicates detailed exploration with examples, L3 involves tutorials or small lab components, L4 integrates homework assignments for reinforcing understanding, and LX entails topics to be explored through Self-Directed Learning (SDL). The first four levels of coverage are facilitated through a combination of lectures (notes and videos), required reading assignments, quizzes, assigned tutorials/labs, and/or homework assignments. In contrast, LX coverage encourages students to explore topics via optional reading assignments, online resources available on the course site, self-identified resources, and self-directed exercises and experimentations. While certain homework assignments also necessitate SDL (e.g., exploring plugin packages not discussed in lectures to implement specific features), we anticipate that the final project provides the most significant opportunity for students to hone their SDL skills.

Topics	Coverage in four levels from lowest to highest: (L1) introductory; (L2) detailed coverage with examples; (L3) with tutorials or small labs: (L4) with homework.				To be explored via SDL (e.g., optional readings and tutorials, some of the homework, open-ended final project).
	L1	L2	L3	L4	LX
Dart Programming Basics	X	X	X	X	
Flutter toolchain setup	X			X	
Flutter Basics (project structure, Widgets and their types, Widget tree, layouts)	X	X	X	X	
Basic Widgets (Row, Column, Container, Icon, Text, Form, Image, Buttons, AppBar, Scaffold, ListView, Stack, SnackBar, AlertDialog, etc.)	X	X	X	X	
Other Widgets (Badge, SliverAppBar, BottomSheet, and many others)					X
Declarative UI and State Management (stateless and stateful Widgets, setState and Provider)	X	X		X	
State Management via other approaches (BLoC, Riverpod, etc.)					X
Navigation in Flutter (navigation between screens, passing data between screens)	X	X		X	
Navigation in Flutter (Tab navigation and drawer navigation)					X
Local data storage (Shared Preferences)	X	X		X	
Local data storage (local databases, e.g., SQLite)					X
Backend cloud storage (Firebase and Firestore)	X		X		X
Flutter Animations	X				X
Flutter and APIs (HTTP requests, working with RESTful APIs, JSON parsing)	X	X		X	
Flutter and Device Features (basic sensors including GPS, permissions handling)	X	X		X	
Flutter and Device Features (other sensors such as camera, accelerometer, etc.)					X
Integrating Google Maps (basics)	X		X		X
Integrating Google Maps (advanced features such as living tracking, camera control, user interaction, etc.)					X
Testing and debugging	X	X		X	X
Mobile security and data privacy					X
Internationalization and Accessibility	X				X
Performance optimization	X				X
Multimedia (audio/video) support					X
Logo and Icon customization					X
Integrate Payment in Flutter					X
Other topics					X

Figure 1: Course topics and coverage.

3.2 Course Material and Resources

With careful consideration, we opted not to adopt a textbook. Instead, aligning with the learning objectives, we developed course material centered around a few example apps we created. This decision was influenced by several observations and considerations:

- The rapid evolution of Flutter and Dart poses challenges for traditional textbooks. Updates in newer editions may lag, rendering content quickly outdated and potentially causing confusion among readers.
- Embracing an example-based learning approach [4] aligns well with the course objectives.
- Google provides robust support for developers through various channels, including a dedicated YouTube channel featuring continually expanding content. This ensures more accurate and up-to-date information compared to static books.
- The burgeoning community of Flutter developers contributes to a wealth of online tutorials and technical articles.

As a result, our course material consists of instructor-created lecture notes and videos, required and optional reading assignments (comprising documentations, articles, tutorials, and videos), Google’s Codelabs, quizzes, and assignments. These materials are delivered to students in the form of weekly modules. Given the extensive range of topics covered, the course heavily depends on online resources. To streamline accessibility, we have established a dedicated webpage to host URLs for a variety of online resources.

3.3 Content Arrangement and Weekly Modules

The course content is presented through weekly modules. Figure 2 (a) displays the content schedule for the entire six weeks, while Figure 2 (b) provides an abridged snapshot exemplifying a week module. In this context, we would like to draw attention to two issues concerning the organization of lectures:

1. Overall, our course adopts an app-driven approach to organize content. The coverage of topics follows a mainline structured around six example apps, progressing from simple to complex. This approach involves a fusion of app-driven lecture videos, where specific apps (e.g., To-Do-List app) are explained, and topic-driven lecture videos, which delve into specific technical aspects (e.g., Flutter navigation).
2. For the app-driven videos, some are crafted in a tutorial style, providing a step-by-step guide to creating an app from scratch, allowing viewers to follow along. Explanations are seamlessly woven into the steps when necessary. This tutorial style is employed for the first three example

Week	Lecture agenda	Assignments and Quizzes	Goals of Week #3:
1	Welcome and introduction. Dart Programming	HW#1, HW#2, HW#3 Course Overview Quiz Dart Quiz#1 Dart Quiz#2	<ul style="list-style-type: none"> Learn to write a small <i>dice game</i>. Gain more experience with stateful Widgets and using <code>setState</code> for state management. Learn to write a <i>to-do-list</i> app. Learn navigation and routing between multiple screens. Learn app state management with <code>Provider</code>. Learn to use <code>SharedPreferences</code> to save data on local devices. Learn to share content in the app via the platform's share dialog. Learn to use various <code>plugin packages</code>. Learn to use <code>ListView</code>, <code>ElevatedButton</code>, <code>TextButton</code>, <code>AlertDialogs</code>, <code>SnackBar</code>, <code>DateTimePicker</code>, and other widgets.
2	Introduction to Flutter; toolchain setup, and <i>Hello World App</i> . Widgets (Stateless vs. Stateful), <i>Layouts</i> ; and <i>Tip Calculator App</i> .	HW#4 and HW#5 Flutter Quiz#1 Flutter Quiz#2	<p>Tasks to Complete for Week #3:</p> <ol style="list-style-type: none"> 1. Watch the lecture video -- "Crap's Game App" (tutorial style video). 2. Complete HW#6 3. Watch the lecture video -- "To-Do-List App: Preview". 4. Watch the lecture video -- "Flutter ListView". 5. Watch Flutter Widget of the Week -- ListView. 6. Read Flutter Doc on ListView Class. 7. Watch the lecture video -- "Flutter Navigation". 8. Read Flutter Doc on Navigation and Routing. 9. Watch the lecture video -- "Flutter Declarative UI and State Management". 10. Read Flutter Doc on State Management. 11. (Optional) Watch "Pragmatic State Management in Flutter" video. 12. Watch the lecture video -- "Flutter Shared Preferences". <ul style="list-style-type: none"> o Asynchronous programming is used when we read/write data with Shared Preferences. You may want to <i>review</i> the following Flutter CodeLab and video about Async coding in Flutter: <ul style="list-style-type: none"> ▪ Dart CodeLab on "Asynchronous programming: futures, async, await". ▪ Flutter in Focus series on <i>asynchronous coding in Dart</i>. 13. Watch the lecture video -- "To-Do-List App: A Closer Look". <ul style="list-style-type: none"> o Additional videos to watch: <ul style="list-style-type: none"> ▪ Flutter Widget of the Week -- AlertDialog. ▪ Flutter Widget of the Week -- SnackBar. 14. Complete HW#7 15. Take Flutter Quiz#3
3	More on Widgets and Basic State Management with <code>setState</code> ; <i>Crap's Game App</i> . ListView, Navigation, State Management with <code>Provider</code> ; Shared Preferences, and <i>To-Do-List App</i>	HW#6 and HW#7 Flutter Quiz#3	
4	Flutter Animations, GestureDetector, CustomPaint, Saving drawing images, Permissions, and <i>Finger-painting App</i> . RESTful web services, HTTP requests, JSON parsing, and <i>Weather App</i> .	HW#8 and HW#9 Flutter Quiz#4 Project Proposal	
5	Firebase and Firestore. Adding Google Maps to a Flutter App. Work on project	HW#10 and HW#11 Project check-point report	
6	Work on Project	Project final report and demo/presentation	

Figure 2: (a) Content schedule in weekly modules; (b) snapshot of module for week #3

apps. However, for the subsequent three apps, characterized by their larger scale, the traditional step-by-step approach would be excessively lengthy and potentially tedious. Consequently, a different strategy was adopted. A concise preview video was created, highlighting the app's functionalities, and enumerating the technical topics embedded within it. This is followed by a series of lecture-style videos, each dedicated to a specific technical topic. Finally, a closer-look lecture video delves into the intricate details of implementing these topics within the app.

The course encompasses 11 homework assignments and one final project. The distribution of the assignments is structured as follows: one on setting up the Flutter development environment and toolchain on the student's personal computer, two on Dart programming, and eight focusing on the creation of apps using Flutter. The final project is intentionally open-ended, affording students the freedom to propose and develop their own mobile app ideas. It provides ample opportunity for students to practice SDL.

3.4 Assessment

Student learning is evaluated through three components: homework assignments, quizzes, and a final project. Given the course's emphasis on practical application over theory, the final project serves as a comprehensive assessment, effectively replacing the final exam. It is introduced as early as the first week, encouraging students to contemplate their project ideas from the course's outset. Project proposals are due midway through the fifth week, and due to the condensed summer schedule, students have only one and a half weeks to complete their projects post proposal submission. A brief check-point report is required by Monday of the sixth week, with the final project submission due on the last day of the sixth week. Additionally, students are expected to share a project presentation/demo video in the course discussion forum. Final project grades are determined based on criteria including novelty, difficulty level, completion status, workload, acquisition of new skills through SDL, and the quality of the accompanying report and presentation/demo.

4 Observations and Feedback

The course garnered positive feedback in all three summer offerings. Except few early withdraws, most students remained actively engaged throughout the entire semester despite the course's noticeable intensity. Students often characterize the course as providing a unique blend of experiences, encompassing excitement, challenges, enjoyment, frustration, and, ultimately, a profound

sense of achievement. Below we summarize the student feedback from their final project reports and course evaluations ¹.

4.1 Evidence of SDL and its Effectiveness

As evidenced by their homework submissions throughout the semester, students have consistently practiced SDL. This commitment to SDL peaks as they embark on completing their final project. Although quite a few students observed their final app deviating somewhat from their original proposal or lacking the full implementation of proposed features, over 90% of students demonstrated, in their final report and demo, the acquisition and application of new skills and techniques in their apps. The following are comments from students, expressing their appreciation for the role of SDL.

- *“This project was both challenging and fun. It challenged me to learn the new skill sets required to build my application, and was fun so I maintained motivation to learn the skills needed to build this project.”*
- *“I really like the idea of trying to figure out code like one would in the real world with research and trial and error. There was a lot that was jam packed into this course and it was really fun learning how to finally make something tangible with code.”*
- *“The course was taught in a way where I had to explore new skills in order to figure out a problem. These skills are used in the real world and it is helpful to practice these skills.”*
- *“Overall, I am fairly pleased with the app that I have created for my final project. I had actually attempted (unsuccessfully) to make this app in Android Studio about 4 years ago based solely on videos and code that I had found on the internet. I feel like this course was well done and prepared me with many of the tools that I needed to make this app.”*

The last comment above is particularly intriguing, prompting reflection on **individual SDL versus SDL under guidance**. For students who are still honing their SDL skills, it seems a certain level of guidance, provided through either a well-structured course or mentorship, can be beneficial.

4.2 Feedback Regarding Course Content and Delivery

Overall, students have expressed favorable opinions about the course structure. While a minority raised concerns about the intensity and workload, the majority of students were well-prepared for the challenges. Here are a few sample comments:

¹The study was reviewed and approved by the Indiana University Institutional Review Board (#19777).

- *“a very logical, well-structured online course that made it easy for me to find resources and manage my time.”*
- *“Excited about the content of the class so everything was more interesting from the students side.”*
- *“The material was challenging but engaging and overall very enjoyable work.”*

The approach we took to compile course materials without using a textbook was well-received. Some students specifically appreciated the tutorial-style videos and the encouragement to explore Google’s official documentation:

- *“The materials for this course were very helpful. Lots of resources were provided, both optional and required, throughout the weeks.”*
- *“I really liked the videos where you walked through how the code was created. I worked right along with you and could pause the video when I had something that I needed to find in the software.”*
- *“Allowing us to look stuff up on our own and go through Flutter’s documentation help me familiarize myself with Flutter relatively quickly.”*
- *“The instructor encouraged us to use the official documentation for Dart and Flutter above other resources and it was really helpful in building my confidence with them.”*

During the 2023 Summer course, a survey was conducted at the end of the semester to gather feedback for further improvements. Due to space constraints, a detailed discussion of all collected data is not feasible. However, we want to emphasize the feedback received on two of the questions: (1) “Among the topics covered in the course, which one do you consider the most challenging?” and (2) “What was the most enjoyable aspect of this course?” Figure 3 visually represents students’ responses to these questions in the form of word clouds.

4.3 Possible Improvements

While there are several areas for potential improvement in the course, two stand out as primary priorities:

1. **Enhance the coverage on Declarative UI and State Management with Provider:** Declarative UI and state management are closely related topics. Given the prevalence of declarative UI in cross-platform frameworks, and its adoption in native mobile development (e.g., Jetpack Compose for Android and SwiftUI for iOS), it has become crucial for mobile developers to navigate the shift from the traditional imperative approach to UI updates. This transition can be challenging to beginners,



Figure 3: Students’ feedback on (a) “the most challenging topic”, and (b) “most enjoyable aspect”.

however, as reflected in students’ feedback. We acknowledge the need to refine our coverage on these topics.

2. **Integrate the topic on Mobile App Architecture:** We are currently exploring ways to incorporate the topic of mobile app architecture (e.g., MVC, MVP, MVVM, and VIPER) into the course. Recognizing the already packed schedule, finding an effective integration approach remains a challenge.

5 Conclusions

Our redesigned Mobile App Development course, centered around Google’s Flutter framework and emphasizing self-directed learning (SDL), has proven to be a dynamic and engaging educational experience over the past three summers. The positive feedback affirms the effectiveness of our SDL approach in cultivating new skills and overcoming challenges. We acknowledge the need for ongoing improvements and expect the course to continue to evolve to meet the ever-changing landscape of mobile app development.

References

[1] G. Altuger-Genc and I. Aydin. “Design and Development of Self-Directed Learning (SDL) Modules for Foundations of Computer Programming Course”. In: *Proceedings of the 122nd ASEE Annual Conference & Exposition*. 2015.

- [2] C. Collier. “Becoming an Autonomous Learner: Building the Skills of Self-Directed Learning”. In: *Journal of Transformative Learning* 9.1 (2022).
- [3] *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022*. URL: <http://www.statista.c> (visited on 01/12/2024).
- [4] P. Deitel, H. Deitel, and A. Wald. *Android 6 for Programmers: An App-Driven Approach*. 3rd. USA: Prentice Hall Press, 2015.
- [5] *Flutter*. URL: <http://www.flutter.dev> (visited on 01/12/2024).
- [6] M. Knowles. *Self-directed learning*. Cambridge University Press, 1975.
- [7] A. Neeman. “Developing a Cross-Platform Mobile Course using a Multi-Paradigm Framework”. In: *Journal of Computing Sciences in Colleges* 37.8 (2022), pp. 11–21.
- [8] M. Rogers and J. Gratch. “A Snapshot of Current and Trending Practices in Mobile Application Development”. In: *Journal of Computing Sciences in Colleges* 37.6 (2022), pp. 54–66.
- [9] S. Sadhukhan. “Fostering Self-Directed Learning Through Student-Question Posing in CS2”. In: *Proceedings of the ACM Conference on Global Computing Education*. Vol. 2. 2023, pp. 181–182.
- [10] L. Zhang, J. Wolfer, and D. Surma. “Reinventing a digital design course: migrating electronics instruction from physics to CS and using problem-based learning”. In: *Journal of Computing Sciences in Colleges* 33.2 (2017), pp. 29–37.

Hack the Border: Empowering Experiential Learning Competencies in Computing through Hackathons *

Christian Servin¹ and Nadia Karichev¹ and J.J. Childress²

¹Computer Science and ITS Program

El Paso Community College

El Paso, TX 79915

cservin1@epcc.edu/nmerzlya@epcc.edu

²Microsoft

Techspark Texas Community

El Paso, TX

Jonathan.Childress@microsoft.com

Abstract

Experiential learning plays a crucial role in education and professional development by allowing individuals to acquire knowledge and skills through direct experience, thereby making learning more engaging and memorable. This hands-on approach fosters critical thinking and problem-solving skills, as learners are encouraged to reflect on their experiences, understand their implications, and apply what they have learned in real-world situations. Competencies gained through experiential learning are highly valued in the workforce because they demonstrate an individual's ability to apply theoretical knowledge effectively in practical scenarios. By emphasizing competencies, educators and employers can ensure that learners and employees are not only knowledgeable but also capable of performing tasks and solving problems efficiently, making

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

them more adaptable and proficient in their respective fields. Recognizing this need, computing competitions, specifically those focused on cybersecurity concepts, commonly referred to as hackathons, have become pivotal. These competitions serve as a platform for experiential learning, enabling students to participate in friendly gamification events. This paper functions as an experiential report, demonstrating the effectiveness of hackathons in enhancing competencies within two-year computing programs at a community college. By fostering experiential learning, these competitions not only enhance the educational experience but also significantly contribute to increased student retention and completion rates in computing programs.

1 Introduction

Hackathons represent time-constrained gatherings where individuals or teams join forces to engage in intensive projects, often centered around software development and creative problem-solving. These events serve as dynamic platforms that cultivate innovation and imagination, enabling participants to delve into novel ideas and emerging technologies. Crucial for honing skills, hackathons provide practical, hands-on experience that enhances both technical expertise and collaborative abilities. By fostering networking and collaboration among a diverse array of participants, these events facilitate the application of acquired knowledge to real-world challenges (refer to [3, 4] for further insight). Moreover, hackathons present valuable recruitment opportunities for companies in search of skilled individuals, contributing to community development by fostering a culture of shared learning and mutual support (refer to [1, 2] for additional perspectives).

While these competitions garner recognition across various academic and industry settings, two-year programs, specifically community and technical colleges, share commonalities with the hackathon vision. **Experiential Learning:** Degrees offered by two-year institutions, including community colleges, are renowned for their emphasis on practical skills, troubleshooting, and hands-on expertise. These programs prioritize skill acquisition, providing students with a robust experiential learning environment. **Industry and Workforce Impact:** Community colleges play a pivotal role in addressing local and regional workforce needs, positioning them as ideal environments for assessing industry requirements and fostering innovations that cater to those needs. **Marketable Skills:** Beyond comprehending technical aspects and solving specific problems, the significance of incorporating “essential soft skills” cannot be overstated. These skills encompass communication, teamwork, problem-solving, time management, empathy, and emotional intelligence, among others. Engaging in activities like hackathons aids in enhancing these marketable

skills, aligning with the evolving demands of industries and organizations that increasingly seek individuals equipped with a blend of technical and interpersonal competencies for roles in high demand, often characterized as “hybrid jobs.”

1.1 Community College Environments

Community and technical colleges, commonly referred to as two-year programs institutions, play a dual role. They not only offer programs designed to facilitate the smooth transfer of students between two-year and four-year institutions but also provide specialized programs to address workforce skills and requirements (e.g., refer to [5, 6, 7]).

2 Hack the Border: An Experience Report

El Paso Community College, in 2021 received the Microsoft grant for the *Skills for Jobs and Livelihoods*, whose intention was to create capacity on cybersecurity awareness and recognize talent and innovation for the region’s needs and demands. The computer science and Information Technology Systems programs developed a series of professional development workshops and an annual hackathon called *Hack the Border: Adversarial Thinking for the Border Good*. The community college is located on the border between El Paso TX and Cd. Juarez Mexico. The competition was intended to recognize issues located in the border region and which solutions can be attracted to industry, public, and private sectors for implementation. There have been two implementations of the series up to now.

2.1 Professional Development Workshops

During the fall semesters, with a special emphasis on October in observance of Cybersecurity Awareness Month, a series of eight workshops is conducted every Thursday. These workshops cover various cybersecurity awareness topics and are open to both the communities of Cd. Juarez and El Paso, TX. For the Fall 2022 semester, the workshop titles included: “Cybersecurity 101” (in Spanish), “Kali Linux: Intro,” “Web Vulnerabilities,” “Package Capturing,” “Penetration Testing,” “DevSecOps Part I and Part II,” and “Intelligence Vulnerabilities.” In the subsequent Fall 2023 semester, the workshop titles were: “Linux Intro,” “Cryptography,” “Web Vulnerabilities,” “Forensics,” “Exploitation (Metasploit) Part I,” and “Part II.” These workshops are facilitated by professionals, faculty members, or trained coaches from the community. Furthermore, these workshops delve into subjects typically not addressed in conventional courses, providing supplementary education that underscores specific topics.

2.2 Binational Hackathon

The hackathon adopts a gamified structure, enabling participants to amass points through diverse avenues known as “villages,” tailored to their individual interests and strengths. Aligned with selected topics from professional development workshops, these villages present a substantial opportunity for workshop attendees to accrue points throughout the competition. The primary objective entails crafting an artifact—such as an infographic, social media video, or a website—that champions cybersecurity awareness in the region (50 points). In addition, the villages feature specific challenges: *Adversarial Thinking (20 points)*: A strategic scenario prompts participants to devise optimal solutions for critical situations in the region. *Capture the Flag (10 points)*: Involves Linux-based challenges. *Secure Code Challenges (10 points)*: Presents Java code snippets with potential bugs, inviting participants to identify and enhance the code. *Forensics Village (10 points)*: Simulates a recovery scenario, requiring participants to utilize appropriate software and methodologies to solve the presented problem.

2.3 The Ultimate and Inclusive Three-day Configuration

The hackathon unfolds over a three-day competition. A detailed structure of the hackathon is shown in Figure 1.

Day 0: Professional Development Day. The initial day, known as day-0, is dedicated to orientation, professional development, and team matching. Participants typically arrive with pre-formed teams, but for those seeking involvement without a team, day-0 serves as an opportunity for team matching and socializing. The official commencement of the hackathon takes place at 1:00 p.m., featuring an hour of social time with puzzles, table games, and other engaging activities. This time also provides a platform for vendors to showcase their products, collect resumes, and promote local job opportunities. The subsequent hour is devoted to a professional panel, where local cybersecurity experts share motivational testimonies emphasizing the importance of engaging in extracurricular activities within the industry. The final two hours of day-0 are divided into three distinct technical workshops. Participants can choose presentations based on their interests, allowing for a tailored and informative experience.

Day 1: Embarking on the Official Hackathon Journey. The inaugural day kicks off with inspiring talks, featuring prominent figures such as the associate president of the college and a distinguished industry or research guest. Teams engage in a day-long immersion into the hackathon theme. Simultaneously, participants can explore various thematic “villages” to accrue points. The in-

corporation of multiple villages running concurrently provides participants the flexibility to attend, fostering leadership and delegation skills. The diverse villages include *adversarial thinking*, *capture the flag*, *secure code challenges*, and *forensics challenges*.

Day 2: Showcase and Commendation. On the concluding day, participants finalize their projects by submitting them to a repository by 7:00 a.m. Sharp at 9:00 a.m., they embark on a concise five-minute presentation before a panel of judges. These presentations are open not only to the community at large but also to students from the local community college. Post presentations, the judges convene to deliberate on the outcomes, amalgamating them with the cumulative points from the diverse villages on Day 1. Subsequently, a ceremonious event unfolds, bestowing awards upon the first, second, and third-place winners. An additional accolade, the Career and Technical Education (CTE) Award, is presented to the team that demonstrates the most significant impact on addressing technical needs in the region.

Thursday 11/16				Friday 11/17			Saturday 11/18		
8:00 AM				Light Breakfast			Light Breakfast		
9:00 AM				Welcome to Day Two Blayne Pritchard, VP Workforce & Continuing Education Myrha Pagel, Ph.D., Dean of Education and Career and Technical Education					
10:00 AM				Hackathon 10:00 am – 5:30 pm			Hackathon Themes: 1. Cybersecurity Awareness and Education 2. Digital Privacy and Online Safety 3. Emerging Computing Technologies		Team Presentations
11:00 AM				Lunch 12:00 p.m. – 1:00 p.m.			Adversarial Thinking Challenges: 1. Misinformation 2. Scams 3. Industrial Systems		VV AST 150
12:00 PM				Social Time			Capture the Flag Challenge VV AST 150		VV AST 150
12:30 PM				Cyber Professional Panel by AST 150			Secure Code Challenge VV AST 150		Hackathon Closing Ceremony and Awards VV AST 150
1:00 PM				SQL Injection Workshop by Sebastian Oufroune VV AST 307			Forensics Challenge VV AST 150		
1:30 PM				Privilege Escalation Workshop by Jonathan Martnez VV AST 346			VV AST 150		
2:00 PM				Dragon City - Preventing a Hacker Disaster Workshop by Oscar Perez, Ph.D. VV AST 150			VV AST 150		
2:30 PM				Registration/Check-in Team-Building Ice Openings Hackathon Hands-on Vendors			VV AST 150		
3:00 PM	VV AST 150			VV AST 150					
3:30 PM	VV AST 150			VV AST 150					
4:00 PM	VV AST 150			VV AST 150					
4:30 PM	VV AST 150			VV AST 150					
5:00 PM	VV AST 150			VV AST 150					
	QA / Closing first day			VV AST 150					

Figure 1: A Three-day Hackathon Agenda

3 Discussion

Community colleges in the United States offer a wide array of technical degrees, encompassing associates, certificates, and valuable credentials such as occupational skills awards. To supplement these educational offerings, hackathons serve as a valuable tool for skill enhancement beyond the scope of traditional coursework. Furthermore, these hackathons play a pivotal role in augmenting marketable skills within the community, providing participants with opportunities to showcase their talents in real-world scenarios.

References

- [1] Tina Chan et al. “Post-Hackathon Learning Circles: Supporting Lean Startup Development”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI EA '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–8. ISBN: 9781450368193. DOI: 10.1145/3334480.3375216. URL: <https://doi.org/10.1145/3334480.3375216>.
- [2] Frank J. Frey and Michael Luks. “The Innovation-Driven Hackathon: One Means for Accelerating Innovation”. In: *Proceedings of the 21st European Conference on Pattern Languages of Programs*. EuroPlop '16. Kaufbeuren, Germany: Association for Computing Machinery, 2016. ISBN: 9781450340748. DOI: 10.1145/3011784.3011794. URL: <https://doi.org/10.1145/3011784.3011794>.
- [3] Alexander Nolte. “Touched by the Hackathon: A Study on the Connection between Hackathon Participants and Start-up Founders”. In: *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-Ups, Platforms, and Ecosystems*. IWSiB 2019. Tallinn, Estonia: Association for Computing Machinery, 2019, pp. 31–36. ISBN: 9781450368544. DOI: 10.1145/3340481.3342735. URL: <https://doi.org/10.1145/3340481.3342735>.
- [4] Alexander Nolte, Irene-Angelica Chounta, and James D. Herbsleb. “What Happens to All These Hackathon Projects? Identifying Factors to Promote Hackathon Project Continuation”. In: *Proc. ACM Hum.-Comput. Interact.* 4.CSCW2 (Oct. 2020). DOI: 10.1145/3415216. URL: <https://doi.org/10.1145/3415216>.
- [5] Christian Servin et al. “Curricular and Pedagogical Considerations in Computer Science Education: The Role of Community Colleges for the Next Decade”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA*. 2024. ISBN: 9798400704239. DOI: 10.1145/3626252.3630819.
- [6] Christian Servin. “Fuzzy Information Processing Computing Curricula: A Perspective from the First Two-Years in Computing Education”. In: *Explainable AI and Other Applications of Fuzzy Techniques: Proceedings of the 2021 Annual Conference of the North American Fuzzy Information Processing Society, NAFIPS 2021*. Springer. 2022, pp. 453–460.
- [7] Cara Tang. “Community College Corner Community colleges in the United States and around the world”. In: *ACM Inroads* 8.1 (2017), pp. 21–23.

Designing a Design-Oriented Course for CS Majors*

Fahmida Hamid
New College of Florida
Sarasota, FL
fhamid@ncf.edu

Abstract

Considering the need for CS students to be exposed to industry-level software design skills and earn the maturity to program to abstraction, an intermediate-level, design-oriented course before Software Engineering may benefit the (novice) programmers in grasping the essence of Software Engineering smoothly. In this article, we present the topics and teaching methodology of such a course, Object-Oriented Design (OOD). The core objective of the course is to prepare students to write modular, maintainable, and robust software solutions using an object-oriented approach. Along with introducing basic design principles and exploring well-known design patterns, we help students polish soft skills highly valued in the industry: creative and critical thinking, teamwork, accountability, integrative learning, inquiry, and analysis. Since recent resources to teach such a course are not widely available, this experience report will benefit instructors interested in preparing such materials.

1 Introduction

As is standard at many institutions, the core of our undergraduate CS curriculum includes Object-Oriented Programming (OOP) and Software Engineering (SE); the OOP course introduces object-oriented principles to students with

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

less than a year of programming experience, and the SE course includes a 10-week team project that asks students to interface with an external stakeholder to solve a real-world problem in our school. The challenge of building an effective bridge between these two courses arises from the vastly different goals of OOP and SE, some of which are addressed below:

In OOP and mostly all other intro programming classes, the emphasis is on teaching the language syntax and writing a correct and efficient solution to a specific problem, which are essential, but we don't always get the scope to guide them to write generalized solutions for handling a group of similar problems. Students rarely reuse existing solutions (or don't write reusable solutions) to similar problems from different contexts unless we assign them such tasks.

As per the ACM 2013 curriculum guideline [5], below are some critical points [page 180] difficult to cover within the SE course only:

- System design principles: Different levels of abstraction, separation of concerns, coupling, and cohesion, and reuse of standard structures
- Structural and behavioral models of software designs
- Refactoring designs using design patterns
- The use of components in design: component selection, design, adaptation, and assembly of components, components and patterns, components and objects
- Maintaining design qualities: redundancy, usability, maintainability, portability

A dedicated course on object-oriented design (OOD) can provide a platform to experience the path to resolving these above issues to some extent. In the following sections, we introduce the topics we include in this course and the techniques we use to challenge students to apply these topics to software problems that arise in practice.

2 Course Organization

This class covers topics at many levels of abstraction: high-level software design principles, including SOLID [6], KISS (Keep It Simple, Stupid), Separation of Concerns, and Principle of Least Knowledge, to name a few, and around fourteen to sixteen well-known design patterns (Figure:1). We discuss case studies illustrating abstract design principles, concrete examples of design pattern usage, and Unified Modeling Language (UML) applications. In preparing this course, we have drawn upon two essential textbooks [3, 4].

We hold two eighty-minute classes and one workshop per week. Students work with peers during the workshop on experiments that interweave programming and design elements. Students also complete three individual as-

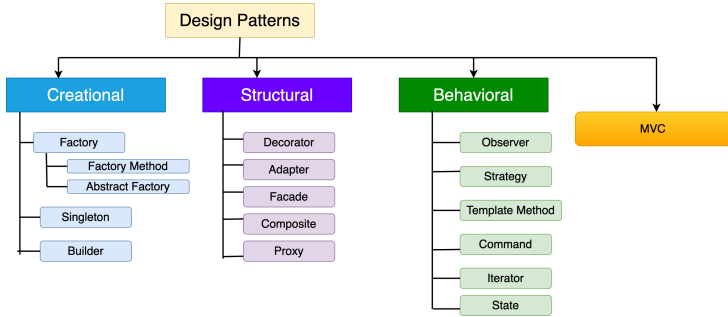


Figure 1: The set of design patterns we usually teach

signments: two heavily focused on design-related issues and one on implementation. At the end, they complete a three-week open-ended final project.

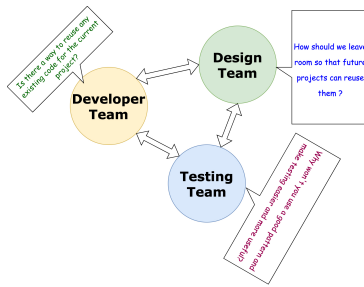
3 Teaching Mechanics: “Explore → Discover → Apply”

During this workshop, students play the roles in three teams (designer, developer, and testing), by rotation, working on the same problem with specific objectives (Figure:2a). The problem description walks them through five scenes to show what may go wrong if proper communication, well-thought designs, and the appropriate patterns were not placed before developing the project (Figure 2b).

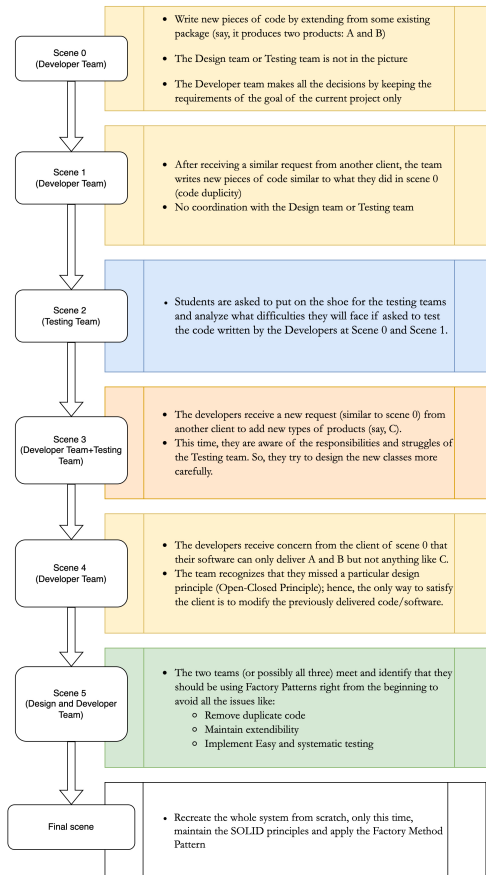
4 Rehearsing Soft-skills

We use bi-weekly assignments (individual) and the final project (group work) to rehearse targeted soft skills. Each assignment narrates a short story like a fairy tale, requiring the students to extract relevant information and develop a tentative design to implement into a computer program. The assignment asks students to design a solution rather than code (solving the problem from an abstract level). Due to page limits, we won’t include any sample of such assignments.

The project requires defining a problem of interest, finding and justifying three patterns that fit the problem context, designing a solution, and applying the chosen patterns in the solutions to assess students’ comprehension of design patterns. The project requires both design and coding/implementation. The overall performance depends on the class presentation (25%), coding with proper documentation (25%), a technical report (25%), and completing a



(a) Collaboration and Communication



(b) Feeling the pains of bad decisions

Figure 2: “You only appreciate a pattern once you have felt this design pain” [2]

questionnaire(25%) at the end. Through the questionnaire, we intend for them to reflect on their behavior and teamwork aptitude.

4.1 Project Setup and The Self-Assessment Questionnaire

Each team elects a team leader. The team leader distributes the tasks among the teammates and schedules the activities and meetings. Team members follow the instructions and communicate with each other as required. After the final submission and presentation, each student answers to the following questions:

A. Team member [except the leader]

- What were the determining factors of choosing your leader?
- Did the leader distribute the tasks in a balanced way? Did he/she consider your strengths and weaknesses before assigning you the tasks?
- Did the leader use any strategy for task distribution and achieving the final goals? State your thoughts.
- Were you comfortable working under the chosen leadership? Why or why not?

B. Team leader

- What measures did you take to ensure the team members follow your lead?
- Did you communicate with the team members before making any decision? Justify your stand.
- Did they provide suggestions or offer alternative solutions to the team on different matters? Briefly explain one such incidence.
- Were your mates respectful towards your leadership?

C. Every Student

- Did you complete your part of the work (design, code, report writing) on time? Explain briefly.
- How did you handle the challenges and criticisms of the overall work? Explain briefly.
- State one strength and one weakness of your team.
- Briefly tell us the best lesson you learned from this teamwork.

The questionnaire is handed to them at the beginning of the project team assignment day, and students are assured that their overall performance will not be impacted by their responses as long as they answer each question reasonably (at least two to three complete sentences).

5 Final Notes

The challenge of offering a design-focused course is that the examples we present to demonstrate the power of design patterns lack the “street cred”; on the other hand, real-world examples, though valuable, are generally too complex to present directly to students [1]. Nonetheless, in the workshop drills, by mimicking the industry-like environment by dividing our students into three teams (design, develop, and test), we attempt to demonstrate the need for collaboration, communication, and the importance of application of design principles and patterns. If offering a full-credit OOD course is not possible in a CS program, we may include some topics and teaching mechanics discussed here in the OOP and SE courses to help students understand the necessity of adapting standard design patterns.

References

- [1] Carl Alphonse, Michael Caspersen, and Adrienne Decker. “Killer" killer examples" for design patterns”. In: *ACM SIGCSE Bulletin* 39.1 (2007), pp. 228–232.
- [2] *Artima - How to Use Design Patterns* — *artima.com*. <https://www.artima.com/articles/how-to-use-design-patterns>. [Accessed 10-01-2024].
- [3] Eric Freeman et al. *Head First Design Patterns*. Ed. by Mike Loukides. 1st ed. O’Reilly Media, 2004. ISBN: 0596007124.
- [4] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st ed. Addison-Wesley Professional, 1994. ISBN: 0201633612.
- [5] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013. ISBN: 9781450323093.
- [6] Robert C. Martin. *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Robert C. Martin Series. Boston, MA: Prentice Hall, 2017. ISBN: 978-0-13-449416-6.

FpTracker—A Labware for Teaching Browser Fingerprinting and Privacy Preservation*

Lin Li and Na Li
Department of Computer Science
Prairie View A&M University
Prairie View, Texas 77446
{lilin, nali}@pvamu.edu

Abstract

As a stateless web tracking technology, browser fingerprint has been widely used in recent years by many websites for collecting users' behaviour data. While benefiting the websites for commercial purpose, the data collection may also cause privacy concerns. To increase students' awareness of web tracking while using the web based services, especially browser fingerprinting, this paper discusses the design and implementation of a labware—FpTracker which was developed for teaching browser fingerprint and privacy protection. Through the labware, students can gain a thorough understanding of the essential concepts of web tracking and browser fingerprinting. This labware can be used for both cybersecurity and trustworthy machine learning. It was tested through a student training workshop conducted in summer 2019 and a security class in spring 2022. Surveys results confirmed the effectiveness of the labware and the attainment of the learning objectives.

1 Introduction

Today, with more and more people using web services such as social networks, personal account management, and online shopping, privacy preservation has

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

gained great attention as many of the services can track users' activities and generate a massive amount of data containing personal information. Web tracking happens when visitors browse the Internet. The websites or the third parties collect, store, and share information about visitors' activities. By analyzing users' behaviours, the websites may infer their preferences and provide content that attract the visitors in order to maximize the commercial benefits. The analysis results can also be of interest to other parties like advertisers. On the other hand, if the information is exposed to malicious users, it can lead to privacy disclosure and other fraudulent conduct which may cause financial loss and panic among people. Thus, understanding the fundamental web tracking techniques is critical for better use of the Internet services and protect privacy.

In general, web tracking technologies can be categorized into two groups: *stated tracking* and *stateless tracking*. Usually, the former is done through cookies, and the latter is conducted through browser fingerprinting. Compared with cookies, browser fingerprinting does not require files to be stored at the end user's device. Instead, it will scan the end user's device in real time and collect a set of browser and system attributes by executing a script in the browser and then apply a hash function to the attributes to create a unique identification of the user device [6]. Like biological fingerprint, the identification can be used to recognize visitors with high accuracy when the time span is short. Therefore, browser fingerprinting is more invasive and opaque in tracking the visitors. Although different teaching materials have been developed for students to learn cookie technologies over the past decade, to the best knowledge of the authors, there are still lack of the effective learning materials that enable students to understand browser fingerprinting and gain hands-on skills. Therefore, this lab was developed for teaching the concepts. The goal is to increase students' awareness of web tracking, especially the browser fingerprinting technology.

2 Related Work

Browser fingerprinting was originally used by web analytics services to measure web traffic and discount fraud clicks. As the client-side scripting has gradually been able to collect more diverse information from the end users' devices, security experts started raising privacy concerns. Dated back to 2012, Mowery and Shacham showed that the HTML5 elements could be used for digital fingerprinting web browsers [7]. In recent years, leveraging machine learning technologies, more researchers found that browser fingerprinting can be used as a reliable source to differentiate end users. For example, Vastel et al. found that on average they could track browsers for 54.48 days and 26% of browsers could be tracked for more than 100 days [9]. Gulyas et al. showed that the user's online behavior could be known by checking the installed browser exten-

sions and the visited websites where they logged into [4]. Meanwhile, actively detecting browser fingerprinting has also become a hot research field. For instance, Iqbal et al. used a machine learning based syntactic-semantic approach to detect browser fingerprinting and they found that more than 10% of the top-10K websites presented browser fingerprinting [5].

Privacy education has been acknowledged in the ACM’s Computer Science Curriculum since 2013 [1]. It was further emphasized in 2020 [2]. A team of cross-disciplinary members, including computer scientists, educators, and social scientists, from the International Computer Science Institute (ICSI) and the University of California at Berkeley, developed an online privacy curriculum [8] including ten principles with the purpose of spreading the awareness of protecting online privacy.

Despite the national demand, the development of hands-on materials for teaching privacy is still insufficient. To engage students and help them gain thorough understanding of web tracking, the work described in this paper presents the design and evaluation of a novel labware for making students aware of the privacy issue while browsing the websites.

3 Labware Design

3.1 Scheme and Implementation

The scheme of our web tracking lab is depicted in Figure 1 which reflects a frequently encountered scenario in people’s daily life: when we browse a website, we often see products promoted in the advertisement banner area and the products are similar to what we recently browsed from other websites. So, we cannot stop questioning “*How could the website know what I like?*”

It should be noted that our labware was developed on top of Dr. Wenliang Du’s SEED lab “Web Tracking” [3] which, however, only introduces web cookies. Thus, our labware covers both the stated and the stateless web tracking technologies: *cookies* and *browser fingerprinting*. Since the cookie part attributes to Dr. Du, we focus on the introduction of our browser fingerprinting lab in this paper and skip the cookie part. Interested educators are encouraged to contact the authors for the whole labware, lab manual, and teaching slides.

The browser fingerprinting lab consists of three components, representing the three web parties depicted in Figure 1: (1) several E-commerce websites with a number of goods listed, (2) an online social network (OSN) website, and (3) the advertisement server. The E-commerce websites host different goods information (e.g., appliances, shoes, phones, etc.) and each web page is embedded with a script from the advertisement server. When a visitor browses the products from page to page, the script will track the browsing record and send the information, including the visitor computer’s ID (i.e., fingerprint or

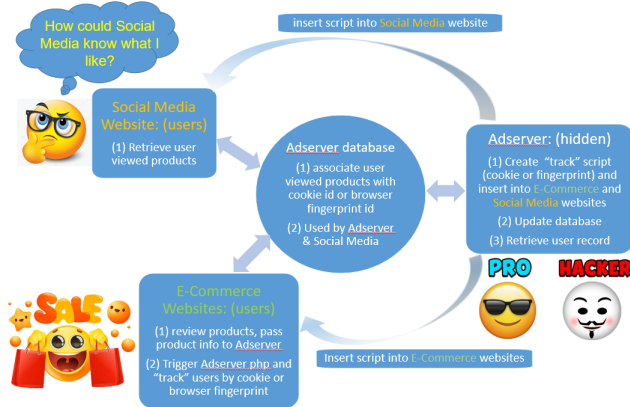


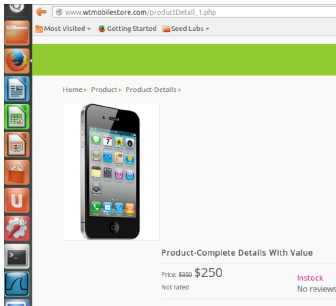
Figure 1: Illustration of Web Tracking

cookie) and the product ID, to the advertisement server. Figure 2 shows an example of the goods listed on the website and the fingerprinting script embedded behind¹. Figure 3 shows the records of fingerprints and associate goods stored in the ad server' database and the script use to connect the database and deposit the records. The OSN website plays a third-party role in which the visitor browses for other purpose (e.g., social networking). Similarly, a script from the advertisement server is embedded into the website. After a visitor browses some goods of the E-commerce websites and then comes to visit the OSN website, the script can retrieve the cookie or fingerprint information and compare it with the historic records stored at the advertisement server. The user's behaviors will be analyzed. A result will be displayed at the OSN website for advertising purpose. The advertisement server is the hub for user information collection, storage, and analysis. Since its scripts are embedded in a hidden mode in the E-commerce and OSN websites, it is invisible to the visitors unless they know how to examine the website source code and how to analyze the network traffic.

3.2 Hands-on Activities and Learning Outcomes

During the hands-on activities, students are required to take the following steps to understand the concepts, observe the results, and study the scripts: (1) visit E-commerce websites and browse the products; (2) log into the MySQL database of the advertisement server, select the database (named "revive_ -adserver"), and open the record table (named "bt_FingerprintLog"). After

¹A JavaScript package fingerprint2.js was used to generate the browser fingerprints.



(a) Goods listed on website

```

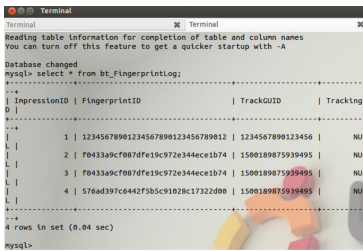
<form name="input" action="productDetail_1.php" method="post">
  <a href="#" class="nav" >imges/logo.jpg" id="mobile"/></a>
  <input class="button" type="submit" value="View Details" /></input>
  <input type="hidden" value="sony" name="nonmobileName" />
  <input type="hidden" value="lcd" name="nonmobileID" />
  <input type="hidden" value="4528402373568" name="idGUID" />
</form>
</div>
</td>
</tr>
</tbody>
</script>
<script src="//fingerprint2.js" ></script>
</script>

var fingerprintReport = function () {
  fingerprint2.get(function(components) {
    var nummur = Fingerprint2.x64hash128(components.map(function (pair) { return pair.value }).join(), 31)
    document.querySelector("#fp").textContent = nummur;
    document.querySelector("#id1").value = nummur;
    document.querySelector("#id2").value = nummur;
    document.querySelector("#id3").value = nummur;
    document.querySelector("#id4").value = nummur;
    document.querySelector("#id5").value = nummur;
    document.querySelector("#id6").value = nummur;
  })
  fingerprintReport();
}
</script>

```

(b) Script of fingerprinting and goods association

Figure 2: E-Commerce websites and embedded scripts



(a) Fingerprint and goods records

```

$com=mysql_connect("localhost","wtuser","seeduser","revive_adserver");
// check connection
if (mysql_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysql_connect_error();
}

// original program mtssed for $fpGUID, should be "".$fpGUID
mysql_query($com,"INSERT INTO bt_fingerprintintlog VALUES (NULL, '',.$fpGUID,'','',.$TrackGu

/*
mysql_query($com,"INSERT INTO bt_fingerprintintlog VALUES (NULL, 'F0433a9cF087f619c972634
mysql_query($com,"INSERT INTO 'bt_fingerprintintlog' (
  'ImpressionID',
  'FingerprintID',
  'TrackGUID',
  'TrackingID'
)
VALUES (
  NULL, ".$fpGUID.", ".$TrackGUID.", NULL
)");

```

(b) Script of fingerprint storage

Figure 3: E-Commerce websites and embedded scripts

selecting each product, they will observe the record change in the table; (3) visit the E-commerce site (“www.wtlabelgg.com”) and observe the product displayed in the banner area; (4) revisit an E-commerce website and open the source code of it to examine the JavaScript code on how the browser fingerprint is generated and how the product information is embedded; (5) open each product page and examine the source code, study how the web uses PHP script to pass the browser fingerprint and product ID to a tracking script at the the advertisement server side; (6) open the source code of the tracking script to examine how the browser fingerprint and product information are inserted into the database; (7) revisit the social network and open its source code to study how the script retrieve the historic records from the advertisement server and display the result based on an analysis of the visitor behaviors. Then, students will be asked to manipulate the scripts to display different private information of the users; and (8) Finally, students will be given a post-lab assignment to investigate the state-of-the-art browser fingerprinting prevention technologies

such as Canvas Poisoner, Agent Spoofer, Tor or BRAVE, etc. A lab report will be due one week after the lab activities.

Through the hands-on activities, students are expected to: (1) be aware of web tracking and its commercial importance and potential threats to user privacy, (2) be able to explain the browser fingerprint mechanism and how to prevent web tracking through browser fingerprint, and (3) be able to analyze web scripts and effectively prevent web tracking.

It is noteworthy that this lab can be used not only for teaching privacy but also for teaching trustworthy machine learning. Once students understand the browser fingerprinting concepts through the lab activities, the instructor can have students analyze a browser fingerprint dataset (e.g., the authors of [9] shared a subset of the browser fingerprint data that they collected through their GitHub account) and apply supervised machine learning algorithms to see on how long the visitors' browser fingerprints do not change and how accurate the browser fingerprints can be used to recognize the visitors.

4 Evaluation

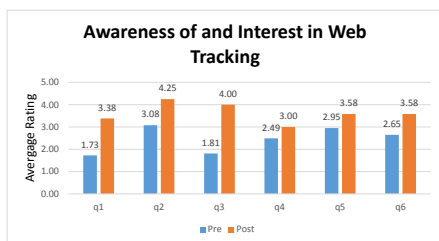
The labware was pilot-tested among student through a summer training workshop in fall 2019. Fourteen undergraduate students majoring in Computer Science participated in the lab evaluation. After the workshop, the authors further refined the labware. In spring 2022², the authors formally evaluated labware within a security course of 23 students. Among the total 37 students, there were 15 girls and 22 boys, including 25 seniors, 4 juniors, 4 sophomores, 3 freshmen, and one graduate student. The learning and evaluation activities fell into two categories: (1) classroom presentation to introduce web tracking basics, and (2) hands-on labs to examine the web tracking using cookies and browser fingerprinting. Pre and post surveys were conducted at the beginning and the end of the workshop to evaluate and analyze the outcomes.

The student survey questions are listed in Table 1. All questions use a rating scale of 1 to 5 with 5 being the greatest deal or the most positive. Questions 1 to 6 were surveyed in both pre and post questionnaires. Figure 4a plots the average rating with regard to the discrepancy of pre and post survey results. A significant increase was observed on students' awareness of and interest in web tracking, browser fingerprinting, and privacy disclosure and preservation after the lab. Questions 7-10 measured how much students gained in understanding the related concepts. The survey results were positive and encouraging with the average rating being greater than 3.6 from all four questions, as shown in Figure 4b. One lesson we learned from the training is to give students more

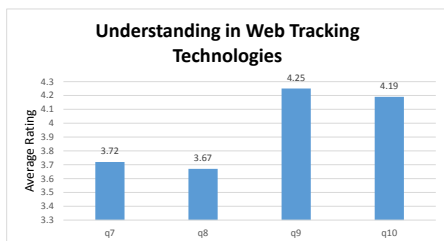
²there was a two-year gap due to the COVID-19 which deferred our evaluation plan

Table 1: Pre and post survey questions

#	Survey Questions	Type
1	Awareness of privacy disclosure and preservation	Pre & Post
2	Awareness about web tracking	Pre & Post
3	Awareness about browser fingerprinting	Pre & Post
4	Interest in privacy disclosure and preservation	Pre & Post
5	Interest in web tracking	Pre & Post
6	Interest in browser fingerprinting	Pre & Post
7	Learning about web tracking and privacy disclosure	Post
8	Learning about browser fingerprinting	Post
9	Understanding different mechanisms of web tracking	Post
10	This lab should be taught in security and privacy courses	Post



(a) Awareness and interest change



(b) Understanding of photo privacy

Figure 4: Experimental results

time³ to study the source codes and figure out the tricks behind the scripts so that they can fully understand the technique details through the activities and provide feedback more positively for Questions 7 and 8.

5 Conclusion

This paper presents the design and implementation of a in-house labware intended for teaching browser fingerprinting in web tracking. The goal is to enable students understand the topic through hands-on activities, which not only make students aware of the possible privacy disclosure via web tracking, but also equips them with necessary defense technologies. After evaluating the labware through different groups of students, the authors received very positive feedback. Students confirmed that they understood the web tracking mechanisms through the lab activities, and the labware helped them learn the concepts effectively.

³We gave students one and a half hours for the lab. Three hours are recommended.

References

- [1] *Computer Science Curricula 2013*. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.
- [2] *Computing Curricula 2020*. <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>.
- [3] Wenliang Du. *SEED*. <https://seedsecuritylabs.org/>. last accessed on August 8, 2022.
- [4] Gabor Gyorgy Gulyas et al. “To Extend or not to Extend: on the Uniqueness of Browser Extensions and Web Logins”. In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. Oct. 2018, pp. 14–27.
- [5] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. “Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors”. In: *Proceedings of IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society. May 2021, pp. 1143–1161.
- [6] Pierre Laperdrix et al. “Browser Fingerprinting: A Survey”. In: *ACM Transactions on the Web* 14.2 (2020), pp. 1–33.
- [7] Keaton Mowery and Hovav Shacham. “Pixel Perfect: Fingerprinting Canvas in HTML5”. In: *Proceedings of W2SP 2012*. Ed. by Matt Fredrikson. IEEE Computer Society. May 2012.
- [8] *Teaching Privacy*. <http://teachingprivacy.org>.
- [9] Antoine Vastel et al. “FP-STALKER: Tracking Browser Fingerprint Evolutions”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 728–741.

Camp CryptoBot: A Model for Taking Risks and Promoting Self-Efficacy in Pursuit of Cybersecurity Career Pathways *

*Pauline Mosley, Li-Chiou Chen, Lisa Ellrodt, and Doris Ulysse Seidenberg School of Computer Science and Information Systems
Pace University
Pleasantville, NY 10570
pmosley@pace.edu*

Abstract

According to the latest Cybersecurity Workforce Study from the International Information System Security Certification Consortium (ISC2) the cybersecurity workforce shortage has risen to a record high of just under 4 million despite the cybersecurity workforce growing by almost 10% in the last year [10]. Only 24% of the U.S. workforce in cybersecurity is female [19]. Furthermore, less than 20% of the students who study cybersecurity in this country are women. Why aren't female students majoring in cybersecurity when this is the fastest growing field? The problem is certainly not ability. Women are certainly as capable as men in succeeding in this field. The gender imbalance has potential consequences for this nation's security, so it is imperative that we understand why women are not being attracted to this field. The problem is that the gap is widening, and we must design and develop innovative ways to engage young women so that we have a diversified and qualified cybersecurity workforce [5]. Our research discusses the Camp CryptoBot model which encourages young women to take risks while combating the

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

fear of failure thereby promoting their self-efficacy. Camp CryptoBot seeks to equip, educate, and empower young women such that they can sustain the rigors of being a woman in a male-dominated discipline and thrive.

1 Introduction

Despite concerted efforts to close the gender gap in computing education and careers, women remain underrepresented in cybersecurity. The US Department of Labor projects that by 2022, 1.2 million computing-related jobs will be created in the US. According to the National Center for Women and Information Technology [2], only about 3% of the available pool of minority high school graduates will earn computing degrees from American colleges and universities, thus lacking the qualifications to fill these jobs. The problem of a low percentage of women applying for the computer science or cybersecurity programs is really a symptom of a much larger problem. The engineering and computing disciplines have been suffering from low female enrollment for a long time. It is interesting to note, that the majority of students in colleges and universities are women, yet the caveat is the percentage of women in computer science is less than 20% of the total student population. In the book entitled *Unlocking the Clubhouse: Women in Computing*, Allen Fisher and Jane Margolis analyze the reasons why women shy away from the computing curriculum's and discovered that one reason is because it is not female-friendly [8]. In another book, *Stuck in the Shallow End: Education, Race and Computing* by Margolis [14], Margolis learned why disadvantaged students in Los Angeles, even when resources are made available to them, do not take advantage of the opportunities as often as other students, because they felt that the computing environment was exclusive and that they did not want to fail. In the famous study conducted by Hill, Corbett, Rose, *Why So Few?* lack of interest and stereotype threat which led to women believing that they would fail at STEM contributed to these women having low self-esteem at STEM courses. Further research shows that student interest in STEM careers is indeed declining [12] [21]. According to the President's Council of Advisors on Science and Technology [16], "... the problem is not just a lack of proficiency among American students; there is also a lack of interest in STEM fields among many students" [1]. This decline appears to be due in part to a lack of awareness of the wide range of STEM (AOS) careers that students can enter with a science background. The need to encourage women and underrepresented minorities to pursue STEM careers is particularly pressing, as their numbers in STEM fields are significantly lower than in the general United States population [4] [3] [20]. This disparity is due, at least in part, to significantly lower completion rates

among underrepresented minorities who pursue STEM degrees [13].

Another factor why there exists of shortage of women in these fields is that women believe that they will not succeed at STEM courses – it is a perceived risk of failing. Perceived Risk of Failing refers high school students’ perception that they are incapable of understanding STEM subjects and will fail if they pursue STEM courses. NSF data revealed that in 2010 only 7.5% of engineering or computer science technicians were African American or Hispanic. This percentage is considerably lower than the 12.2% national population of African Americans and Hispanics [7]. Experts in the field attribute this lack of STEM+C interest to students’ perceptions that computing is “too hard”, “expensive”, and inadequate preparation and encouragement along educational transition points. Student attitudes about their learning play a significant role in their successful acquisition of math skills. There are qualitative studies of students’ perceptions of learning in other disciplines, for example: nursing education, online learning, nutrition, foreign language education, and science education [6] [11] [17] [18] [15].

2 GenCyber Camp CryptoBot

Pace University GenCyber co-ed Camp CryptoBot is an interactive and mission-driven camp that launched in 2016 and has operated 6 times. Camp CryptoBot has introduced a total of 198 high-school students (103 males and 95 females) to the basics of cybersecurity utilizing SeaPerch (underwater robot) and Sphero as the platform for teaching cryptography and cybersecurity concepts using various pedagogical approaches, including storytelling, hands-on labs, and problem-solving missions. The program time-line which entails pre/post and summer camp activities totaling 54 hours is defined in Table 1. It is a seven-day non-residential workshop held from 9am – 4pm with 1 instructional hour and 5 hands-on/group activity hours. Recognizing, the urgent need to expand the workforce pool of cybersecurity professionals of underrepresented groups in the next twenty years, the goal of the CryptoBot summer camp’s mission is both an education and cybersecurity pipeline restoration initiative. It has successfully increased student’s interest, in particular young women, to consider cybersecurity as a career option. In addition, it has provided these students a space where they can take risks and learn how to “fail” thereby increasing their self-efficacy. A major goal of this summer camp is not only to increase awareness of cryptography and cybersecurity concepts using robotics is but to remove all fear of failing and misconceptions about programming and cryptography before one tries, with an emphasis on targeting young women.

The Camp CryptoBot curriculum is a model that utilizes a problem-based cooperative learning approach to subject immersion, idea generation, and group

Table 1: Program Timeline – Pre/Post and Summer Camp Activities

CAMP	Time	Pre-Camp Activities	Hours
PRE MARCH - JUNE	Zoom Meeting	Introductions	1.0
	In Person Meeting	Orientation	2.0
	In Person Meeting	Team Building and Ethics	2.0
	Video To Watch	Introduction to Cybersecurity Careers	1.0
	Video To Watch	Introduction to Programming (Needed for Sphero)	2.0
	In Person Meeting	Introduction to Cybersecurity Concepts	2.0
	Social Media	Ongoing Twitter/Instagram Outreach	2.0
Pre-Camp Outreach Total Hours			12
CAMP CRYPTOBOT July 10- 14	08:30AM-09:00AM	Breakfast and Camp Announcements	1.0
	09:00AM-10:00AM	Ethics	1.0
	10:00AM-11:00AM	Cyber Concepts	.15
	11:00AM-11:15AM	Break – Daily Squawk	1.0
	11:15AM-12:15AM	Cryptography	1.0
	12:15PM-01:00PM	Lunch	.15
	01:00PM-02:00PM	Sphero OR Seaperch Design	.45
	02:00PM-02:10PM	Break – Kahoot's	.45
	02:10PM-02:45PM	Team Nest – Challenge Prep	
	02:45PM-03:15PM	Crypto-Bot Mission – Challenge	
	03:15PM-03:30PM	Mission Status Reports/Kahoot's Reflections	
Summer Camp Instructional (6hrs@ 5 days) Total Hours			30
POST /12# - 9 /31/ 12	In Person On-line	After Camp – What Next?	2.0
	In Person	CISO – Cybersecurity Courses	4.0
	Zoom Meeting In	Cyber Projects – Bring Cyber to Your School	2.0
	Person	Student Cyber Path Advisement	2.0
		Camp Alumni Reunion -	2.0
Post-Camp Outreach Total Hours			12

construction. It facilitates a learner-center classroom because the instructor can tailor pedagogical methodologies for various student learning patterns, and multiple strategies can be implemented to engage all students, regardless of gender or race. We will utilize classroom discussion strategies, kinesthetic activities, and cooperative learning methodologies to address perceived risks of failing and promote self-efficacy. These approaches encourage students to explore and question how and what they are learning as well as take risks - thereby they begin to define their personal expertise, interests, and identity as a learner. These avenues of exploration overlap and merge in organic ways within each child's unique learning experience. Multiple dynamic fun-filled team activities will take place in all the modules which will reinforce cybersecurity first principles and basic network concepts thereby promoting team dynamics and critical thinking skills. Each day students are given an opportunity to think, to take risks and most importantly to believe in themselves (self-efficacy). This is implemented by a series of missions which that are given each day. Students learn how to decode messages using various ciphers (Morse Code, Caesar cipher, vigenere cipher, Binary code, and pig pen cipher) as well as how to send messages using Spheros. The plot of the camp is that Pace University has been hacked, and the students working in teams must commu-

Table 2: Crypto Bot Missions Mapped to Cybersecurity Concepts

#	Crypto Bot Missions	Learning Objectives Mapped to Cybersecurity Concepts
1	To design and construct a Seaperch system with hidden information and a security feature. Students are to explain how many layers in their system.	<ul style="list-style-type: none"> Students will learn how to compare the benefits and challenges of designing a Simple or a Defense in Depth system. Students will be able to describe how layering various ciphers gives depth to defense. Students will comprehend how simple designs are user friendly and easy to troubleshoot but not as secure as in-depth defense systems.
2	To determine how the Seaperch will communicate to the Sphero confidentially.	<ul style="list-style-type: none"> Students will be able to describe why Confidentiality is important for secure communication. Students will be able to illustrate how their design (simple or DID) supports confidentiality in communicating between devices.
3	Demonstrate how the CIA triad is being exhibited in the team's design and communication.	<ul style="list-style-type: none"> Students will understand why the CIA triad is critical to having a secure system. Students will recognize the difficulties in managing information resources. Students will learn the value of having information that is Available and Accurate(Integrity).
4	To send a cryptic message securely using Seaperches, and Sphero's to have a defense strategy in the advent of a cyber-attack either in air or in water.	<ul style="list-style-type: none"> Students will understand the benefits of Thinking Like an Adversary to defend your system accordingly. Students will recognize a systems' vulnerability and the possible damage from malfunctions.
5	To reflect on all missions and analyze the strengths and weaknesses encountered during the missions.	<ul style="list-style-type: none"> Students will recognize how all concepts play a role in cybersecurity. Students will be able to discuss how ethical behavior relates to cybersecurity.

nicate secretly and find clues around the university to determine who hacked the university. Each day they learn a cybersecurity concept and using problem solving techniques coupled with the concept they must complete a mission. The missions are designed to provoke them to take risks, experiment and at times fail. Student, in particularly young women, must learn how to recover from the failure and transition to a place of success. These skill sets are critical for succeeding and sustaining success in cybersecurity. Table 2 depicts the missions and the learning objectives for each day.

3 Data Analyses

A total of 95 girls from 18 high schools participated in Camp CryptoBot from 2016-2023 (camp was not funded in 2017). Our target effort in recruiting Hispanic and African Americans was challenging, and this could be attributed to transportation issues. Figure 1 reveals the ethnicity distribution among female participants. The 22.22% participation for White is the highest of all ethnic groups and 5.56% participation for Black or African American is the lowest. At the end of each camp event, a rubric was shared with the students to help them understand how their learning would be evaluated.

The evaluations were conducted by GenCyber and Dark Enterprises. The Four-Phase Model of Interest Development, by Hidi S. and Reninger K [9] is the framework being used to analyze GenCyber campers' responses to the end of camp survey in an effort to 1) identify and classify the incoming and outgoing interest levels of attendees at the camp, as well as interest development, and 2)

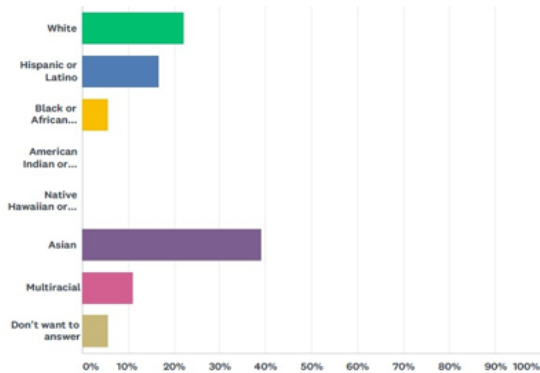


Figure 1: Female Participant Ethnicity Distribution

provide feedback on the interest development efficacy of the camp using seven construct and open-ended items of enablers and barriers. [Dark Enterprises] Though most of the participants had no prior knowledge of cybersecurity or limited coding experience, by the end of the camp, they were able to code Sphero and implement cybersecurity concepts. Participants were asked about their experience at the camp in respect to their interest of cybersecurity. The girls from all six years assessed their level of satisfaction in the five categories.

Figure 2 is a bar chart showing the average incoming and outgoing interest levels by group, and interest change for each of the three groups. As can be seen, the group with low incoming interest had a substantial increase. The moderate group had a modest increase and high incoming interest had an incremental increase. Girls also mentioned how this camp helped them to get excited and pursue their career in cybersecurity. Others mentioned that they had never heard of cybersecurity until this camp. When asked to describe things that happened at camp which enabled interest in cybersecurity, students reported the following:

- Working with Sphero and learning ciphers like morse code

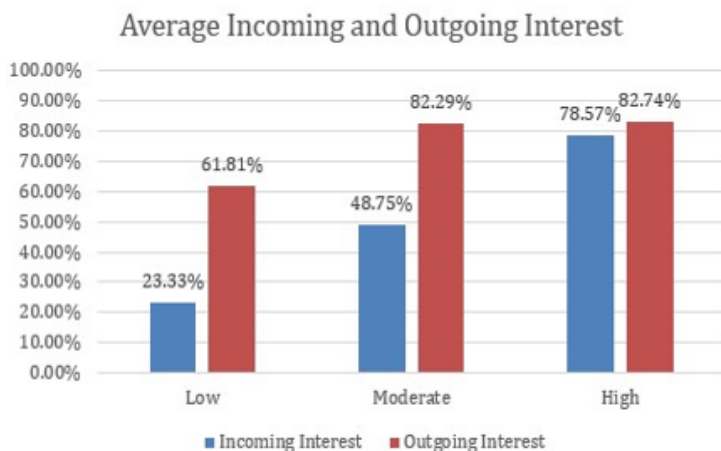


Figure 2: Student Interest

- The coding on my Sphero, the morse code learning, and the decrypting, and encrypting codes. I got to feel like a mini hacker or feel like I was in the matrix.
- Meeting new people from different areas in the world. learning about Caesar cipher. learning how to program the Sphero in many ways. the short games we did with each other. lastly, the challenges we competed in.
- Learning about all the dangers of cyber attacks and how this issue is growing really opened my eyes to how cybersecurity is needed for the future.
- Investigations based on the scenario, how to decode encode, all Cipher drills, lots of enrichment which nourished knowledge at the same time without compromising fun and ethics, new career options
- The team challenges helped solidify what we were taught in lectures
- The various challenges, the guest speakers, the different ciphers, the people I worked with
- Ciphers and the lectures from corporation employees with firsthand experience of cybersecurity
- Sphero Learning the different functions Activities Cyber security cards

- Microsoft speakers, group discussions, and Sphero coding/encrypting
- My team leader was extremely encouraging and helped me open up to my team. Having such a great team therefore led me to participate more and become more interested , the great variety of guest speakers, they covered many different topics and areas so it expanded my knowledge and gave me new interests in parts of cybersecurity that I never knew about
- I got to hear a bunch of people talk about interesting things for about 8 days
- The challenges and how we were rewarded for learning about the cybersecurity concepts taught and the knowledge we brought to the camp.

4 Concluding Discussions

In this paper, we have presented a Camp Cryptobot model for high school students, in particularly girls that was designed to attract a diverse group of girls and engage them in activities that cultivate them to take risks and enhance their self-efficacy leading them toward a cybersecurity career. There are many summer camp experiences for young woman, however, what sets this camp apart is the consideration factors to enhance female self-efficacy and negate female fear of failing. The design of the lessons is created by women for women to promote inclusion and diversity, as well as to introduce and sustain best practices that encourage women to stay in the field. If you can see it you can become it – another key factor of this model is the diversity of female faculty. In 2020, Camp CryptoBot was held virtually and even in the modality, we were able to create an online environment in which the girls felt supported and were inspired to consider cybersecurity. As a result of this model operating for six years, our findings reveal that the girls participating in this camp showed evidence of growth in self-efficacy through their developing abilities in robotics and programming skills. Our model contributes to the evidence that it is possible to recruit and attract a diverse group of young women to cybersecurity.

5 Acknowledgments

Special thanks to the participants, Mr. John Sarlo, the High School Pedagogical Consultant, The United States Navy Personnel, industry experts from IBM, Goggle, MasterCard, and other cybersecurity firms. Thanks to Dark Enterprises for conducting the surveys and assessment for all six camps. This

material is based upon work supported by the National Science Foundation, National Security Agency and GenCyber H98230-22-1-0176.

References

- [1] Ian C Binns et al. “Student perceptions of a summer ventures in science and mathematics camp experience”. In: *School Science and Mathematics* 116.8 (2016), pp. 420–429.
- [2] Tracy Camp, Christine Liebe, and Michelle Slattery. “Applying NCWIT Protocol to Broaden Participation in Computing: A Case Study of CS@ Mines”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 528–534.
- [3] Christianne Corbett, Catherine Hill, and Andresse St Rose. *Where the Girls Are: The Facts about Gender Equity in Education*. ERIC, 2008.
- [4] National Research Council et al. *Gender differences at critical transitions in the careers of science, engineering, and mathematics faculty*. National Academies Press, 2010.
- [5] David Dampier. “Building an education program for engineers in digital forensics”. In: *2008 Annual Conference & Exposition*. 2008, pp. 13–264.
- [6] Dorothy H Evensen, Jill D Salisbury-Glennon, and Jerry Glenn. “A qualitative study of six medical students in a problem-based curriculum: Toward a situated model of self-regulation.” In: *Journal of Educational Psychology* 93.4 (2001), p. 659.
- [7] Richard Fry, Brian Kennedy, and Cary Funk. “STEM jobs see uneven progress in increasing gender, racial and ethnic diversity”. In: *Pew Research Center* 1 (2021).
- [8] Eric Gorki. “College Gender Gap Favoring Women Stops Growing”. In: *The Washington Post* 26 (2010).
- [9] Suzanne Hidi and K Ann Renninger. “The four-phase model of interest development”. In: *Educational psychologist* 41.2 (2006), pp. 111–127.
- [10] *Key Findings from the ISC2 Cybersecurity Workforce Study*. 2024. URL: <https://niccs.cisa.gov/cybersecurity-career-resources/featured-stories/key-findings-isc2-cybersecurity-workforce-study>.
- [11] Teresa Lao and Carmen Gonzales. “Understanding online learning through a qualitative description of professors and students’ experiences”. In: *Journal of Technology and Teacher Education* 13.3 (2005), pp. 459–474.

- [12] B Lindsay Lowell, Hal Salzman, and Hamutal Bernstein. “Steady as she goes? Three generations of students through the science and engineering pipeline”. In: (2009).
- [13] Jane Margolis. *Stuck in the shallow end, updated edition: Education, race, and computing*. MIT press, 2017.
- [14] Jane Margolis and Allan Fisher. *Unlocking the clubhouse: Women in computing*. MIT press, 2002.
- [15] Michael Prelip et al. “Role of classroom teachers in nutrition and physical education”. In: *Californian Journal of Health Promotion* 4.3 (2006), pp. 116–127.
- [16] *Prepare and inspire: K-12 education in science, technology, engineering and math (STEM) for America’s future*. 2010. URL: https://www.nsf.gov/attachments/117803/public/2a--Prepare_and_Inspire--PCAST.pdf.
- [17] Kimberly Starr and Virginia M Conley. “Becoming a registered nurse: The nurse extern experience”. In: *The Journal of Continuing Education in Nursing* 37.2 (2006), pp. 86–92.
- [18] Brad Sullivan and Edwin M Everham. “Life at the Boundaries: Exploring Interdisciplinarity in an Environmental Literature Course”. In: *Interdisciplinary literary studies* 5.2 (2004), pp. 1–15.
- [19] *Women Represent 24 Percent of Cybersecurity Workforce, (ISC)² Reports*. 2019. URL: <https://www.securitymagazine.com/articles/90071-women-represent-24-percent-of-cybersecurity-workforce-isc-reports>.
- [20] *Women, minorities, and persons with disabilities in science and engineering: 2013. Special Report NSF 13-304*. Arlington, VA. 2013. URL: <http://www.nsf.gov/statistics/wmpd/>.
- [21] Vanessa L Wyss, Diane Heulskamp, and Cathy J Siebert. “Increasing middle school student interest in STEM careers with videos of scientists.” In: *International journal of environmental and science education* 7.4 (2012), pp. 501–522.

The Utility of Radix Representations and Surrogate Logarithms in the Analysis of Algorithms and Data Structures *

Michael Kart
Department of Computer Sciences
Saint Edward's University
Austin, TX 78704
michaelkart@stedwards.edu

Abstract

Students often encounter the concepts of algorithmic complexity before they have fully developed calculus-level mathematical sophistication and their ability to handle abstraction. Undergraduate computer science students tend to struggle with fundamental complexity classes especially when logarithms are involved. This paper proposes the use of radix representations and a simple surrogate for logarithms in order to improve student understanding of the complexity analysis of selected algorithms.

1 Introduction

Students taking an algorithms and data structures course are guaranteed to encounter the concepts of asymptotic analysis, complexity classes, and algorithms. Unfortunately, many students have gaps in their mathematical preparation and it is incumbent on the instructor to mitigate this problem. Since a full gap coverage is not possible, instructors should leverage strong student understanding of related topics. We propose using radix representations along with a simple surrogate for logarithms to help fill some of these gaps since

*Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

activating prior knowledge in students that is both relevant and congruent has a positive effect on learning gains[1]. Note that these tools are intended to scaffold student understanding and perhaps act as a gateway to a deeper understanding of the more rigorous techniques of differential calculus.

This paper proceeds by providing content that is relevant to these ideas. A background is provided on radix representations and the preferred definition of asymptotic notation for the course. It is important to note that students have already been introduced to radix representations in a previous course. A unique introduction can be found in [3]. It should also be noted that selected course content follows, but this content is not contiguous since algorithmic (time) complexity discussions occur throughout the course. Instead, content is presented as particular data structures are introduced and studied.

2 Background

2.1 Radix Representations

Let n be a nonnegative integer and b be an integer that is greater than 1. Then, for some integer $k \geq 0$, n can be uniquely decomposed over the nonnegative powers of b as follows:

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \dots + r_0 \cdot b^0, \tag{1}$$

where $0 \leq r_i < b$ for all $i \in [0, k]$ and $0 < r_k$. The numeral $r'_k r'_{k-1} r'_{k-2} \dots r'_0$ is defined to be the representation of n in radix (i.e., base) b , where r'_i is the digit that represents the integer r_i , for each $i \in [0, k]$.

For example, with $n = 98$ and $b = 2$:

$$98 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0. \tag{2}$$

Therefore, the radix 2 representation of 98 is 1100010.

2.2 Asymptotic Notation

Definition 1. Let $f(n)$ and $g(n)$ be functions over the positive integers. We write $f(n) = O(g(n))$ if there exists a real number c and positive integer n_0 such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. Additionally, we write $f(n) = \Theta(g(n))$ when both $f(n) = O(g(n))$ and $g(n) = O(f(n))$ hold.

Further details can be found in [2].

3 Introduction to Algorithm Analysis

Find the minimum integer in the following list of size seven:

304788
660215
271370
833580
56277
410706
709822

As expected, all students are able to find the answer. More importantly, when asked what their algorithm was, students immediately start to discuss the number of digits in each integer and point out that 56277 is the only integer in the list that has 5 digits, while the other integers all have 6 digits. Their algorithm then correctly concludes that 56277 is the minimum. The class is then asked to consider whether this algorithm “inspects” each and every integer in the list and how many steps each inspection takes.

Find the minimum integer in the following list of size seven:

56277
271370
304788
410706
660215
709822
833580

The question is then changed to “What if the list was sorted in non-decreasing order?” Discussion then focuses on whether this is known as a precondition or not. Finally, it is shown that when the list is guaranteed to be sorted in non-decreasing order and contains at least one element, then the algorithm that produces the minimum by simply returning the integer at index 0 is $O(1)$.

3.1 Alternative to Logarithms

We now introduce the function $len_b(n)$, which almost every student grasps immediately, as a surrogate for the normal logarithm.

Definition 2. Let $b > 1$ and $n \geq 1$ both be integers. The $len_b(n)$ is defined to be the length of the shortest representation of n in radix b (i.e., the representation that does not include leading zeros). So, $len_b(n) = k + 1$ when:

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \dots + r_0 \cdot b^0, \quad (3)$$

where $0 \leq r_i < b$ for all $i \in [0, k]$ and $0 < r_k$.

The normal logarithm from math is defined as follows:

Definition 3. Let $b > 1$ and $n \geq 1$ both be integers. Then $\log_b(n) = x$, where $b^x = n$.

The following lemma shows that the functions len_b and \log_b never differ by more than 1.

Lemma 1. Let $b > 1$ and $n \geq 1$ both be integers. Then

$$\text{len}_b(n) - 1 \leq \log_b(n) < \text{len}_b(n). \quad (4)$$

Proof. Suppose $k + 1 = \text{len}_b(n)$. Then we want to show

$$k \leq \log_b(n) \leq k + 1 \quad (5)$$

since $\text{len}_b(n) - 1 = k$ and $\text{len}_b(n) = k + 1$. By definition of $\text{len}_b(n)$, we have:

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \dots + r_0 \cdot b^0 < b^{k+1} \quad (6)$$

Notice that $r_k \cdot b^k \leq n$ since $b \geq 1$ and $r_i \geq 0$ for all $i \in [0, k]$ implies that $r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \dots + r_0 \cdot b^0 \geq 0$. Next, observe that $b^k \leq r_k \cdot b^k$ since $1 \leq r_k$. Therefore, $b^k \leq n < b^{k+1}$ and, consequently, $k = \log_b(b^k) \leq \log_b(n) \leq \log_b(b^{k+1}) = k + 1$ since \log_b is an increasing function and $\log_b(b^m) = m$ for any m , which follows directly from the definition of \log_b . So, we have established Equation 5, as desired. \square

Next, we show that len_b and \log_b are in the same complexity class.

Theorem 1. Let $b > 1$ be fixed. Then $\log_b(n) = \Theta(\text{len}_b(n))$.

Proof. By Lemma 1, we have:

$$\text{len}_b(n) - 1 \leq \log_b(n) < \text{len}_b(n). \quad (7)$$

This implies that $\frac{\log_b(n)}{\text{len}_b(n)} < 1$ for all n , so $\log_b(n) = O(\text{len}_b(n))$. It also implies that $\frac{\text{len}_b(n)-1}{\log_b(n)} \leq 1$. So, we have $\frac{\text{len}_b(n)-1}{\log_b(n)} = \frac{\text{len}_b(n)}{\log_b(n)} - \frac{1}{\log_b(n)} \leq 1$ and $\frac{\text{len}_b(n)}{\log_b(n)} \leq 1 + \frac{1}{\log_b(n)}$. Observe that, by definition, $\log_b(b^1) = 1$ and $1 \leq \log_b(n)$ for $n \geq b^1 = b$. For $n \geq b$, $\frac{1}{\log_b(n)} \leq 1$, and $\frac{\text{len}_b(n)}{\log_b(n)} \leq 1 + (\frac{1}{\log_b(n)}) \leq 1 + (1) = 2$ for $n \geq b$. So, $\text{len}_b(n) = O(\log_b(n))$. Since $\log_b(n) = O(\text{len}_b(n))$ and $\text{len}_b(n) = O(\log_b(n))$, we have shown that $\log_b(n) = \Theta(\text{len}_b(n))$, as desired. \square

In the following lemma, we show that the function len_b doesn't increase rapidly.

Lemma 2. *Let the integer $b > 1$ be fixed. Then, for $n > 0$,*

$$\text{len}_b(n) \leq \text{len}_b(n+1) \leq \text{len}_b(n) + 1. \quad (8)$$

Proof. Suppose $k = \text{len}_b(n)$. Then

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \dots + r_0 \cdot b^0, \quad (9)$$

where $0 \leq r_i < b$ for all $i \in [0, k]$ and $0 < r_k$. It follows that $b^k \leq n$, which implies that $b^k \leq n+1$ and, consequently, $\text{len}_b(n) = k \leq \text{len}_b(n+1)$.

Furthermore, since each $r_i < b$,

$$\begin{aligned} n &\leq (b-1)(b^k + b^{k-1} + b^{k-2} + \dots + b^0) \\ &= b(b^k + b^{k-1} + b^{k-2} + \dots + b^0) - 1(b^k + b^{k-1} + b^{k-2} + \dots + b^0) \\ &= b^{k+1} + b^k + b^{k-1} + \dots + b^1 - b^k - b^{k-1} - b^{k-2} - \dots - b^0 \\ &= b^{k+1} + (b^k - b^k) + (b^{k-1} - b^{k-1}) + \dots + (b^1 - b^1) - b^0 \\ &= b^{k+1} - b^0 \\ &= b^{k+1} - 1. \end{aligned}$$

It follows that $n+1 \leq b^{k+1}$, which implies that $\text{len}_b(n+1) \leq k+1 = \text{len}_b(n)+1$ since each coefficient of b^m is 0 for any $m > k+1$.

So, we have shown both $\text{len}_b(n) \leq \text{len}_b(n+1)$ and $\text{len}_b(n+1) \leq \text{len}_b(n)+1$, as desired. \square

Finally, we show how len_b relates to n .

Theorem 2. *Let the integer $b > 1$ be fixed. Then $1 \leq \text{len}_b(n) \leq n$ for $n \geq 1$.*

Proof. By induction. Let $P(k)$ be the predicate that $1 \leq \text{len}_b(k) \leq k$ for $k \geq 1$.

Base Case ($k = 1$):

$P(1)$ is the predicate $1 \leq \text{len}_b(1) \leq 1$, which holds because $\text{len}_b(1) = 1$.

Induction Step: Let $k \geq 1$ be fixed and assume that $P(k)$ holds, that is, $1 \leq \text{len}_b(k) \leq k$. Now, show that $P(k+1)$ holds. From Lemma 2, we have that $\text{len}_b(k) \leq \text{len}_b(k+1) \leq \text{len}_b(k) + 1$. Combining this with the induction hypothesis, we have that $1 \leq \text{len}_b(k) \leq \text{len}_b(k+1) \leq \text{len}_b(k) + 1 \leq k + 1$, which implies $1 \leq \text{len}_b(k+1) \leq k + 1$, as desired. \square

3.2 Applications

3.2.1 Complexity Class Ordering

Students often guess at whether $\log_b(n)$ or n is smaller. However, students don't have the same problem when comparing $\text{len}_b(n)$ and n .

Let the integer $b > 1$ be fixed. From Theorem 2 it follows directly that

$$O(1) \subseteq O(\text{len}_b(n)) \subseteq O(n). \quad (10)$$

Notice that by multiplying each part of the inequality of Theorem 2 by n results in the inequality $n \leq n \cdot \text{len}_b(n) \leq n^2$, which yields:

$$O(n) \subseteq O(n \cdot \text{len}_b(n)) \subseteq O(n^2). \quad (11)$$

3.2.2 Analysis of Binary Search

The following is a typical Java implementation of the binary search algorithm (see [4]), with the exception that the length of the search interval has been made explicit via the variable named `lengthInterval`:

```
1 public static final int NOT_FOUND = -1;
2 //part of post: rv != NOT_FOUND <==> intArray[rv] == target
3 //part of post: rv == NOT_FOUND <==> intArray[i] != target,
   \forall i \in [0, intArray.length]
4 public static int binarySearch(int[] intArray, int target)
5 {
6     assert intArray != null : "intArray is null!";
7     assert isSortedNonDecreasing(intArray) : "Not sorted!";
8
9     int lowIndex = 0;
10    int highIndex = intArray.length - 1;
11
12    boolean foundTarget = false;
13    int lengthInterval = (highIndex - lowIndex + 1);
14    while(!foundTarget && lengthInterval > 0)
15    {
16        int middleIndex = lowIndex + (highIndex - lowIndex)
17            /2;
18        int middleValue = intArray[middleIndex];
19        if(middleValue == target)
20            foundTarget = true;
21        else
22        {
23            if(middleValue < target)
24                lowIndex = middleIndex + 1;
```

```

24         else //middleValue > target
25             highIndex = middleIndex - 1;
26             lengthInterval = (highIndex - lowIndex + 1);
27     }
28 }
29 assert !(foundTarget && intArray[middleIndex] != target)
30 ;
31 return (foundTarget ? middleIndex : NOT_FOUND);
32 }

```

The worst case running time occurs when `target` is not found. For those cases, the terminating condition for the algorithm becomes (`lengthInterval > 0`). Since `lengthInterval` changes once per iteration of the while loop (assuming `foundTarget` remains `false`), we only need to analyze the sequence lengthInterval_i , which is defined to be the value of `lengthInterval` at the time of its i^{th} update. Note that, by definition, lengthInterval_0 equals `intArray.length`. Define the sequence middleIndex_i in the analogous manner. Lastly, define lowIndex_i and highIndex_i to be the values of `lowIndex`, and `highIndex`, respectively, at the time of the i^{th} update to either of `lowIndex` on Line 23 or `highIndex` on Line 25.

Example 1. Suppose that the following call is made:

$$\text{binarySearch}([1, 2, 4, 8, 16, 32, 64, 128, 256, 512], 999).$$

Then we have:

i	length_i	lowIndex_i	middleIndex_i	highIndex_i
0	10	0	4	9
1	5	5	7	9
2	2	8	8	9
3	1	9	9	9
4	0	10	-	9

The sequence length_i is 10, 5, 2, 1, 0. Rewriting this sequence using the radix 2 representation for each element yields the sequence 1010, 101, 10, 1, 0 and observe that $\text{length}_k = (\text{length}_{k-1} \gg 1)$ for each $k > 1$, where \gg is the Java signed binary number right shift operator. It follows that the for loop is fully executed $\text{len}_2(10) = 4$ times.

While not presented here, it can be shown that when binary search is called with a `target` value is greater than anything in the array, the sequence lengthInterval_k satisfies $\text{lengthInterval}_k = (\text{lengthInterval}_{k-1} \gg 1)$ for each $k > 1$ such that lengthInterval_k is defined. Therefore, the worst-case running time is $O(\text{len}_2(n))$, where $n = \text{intArray.length}$.

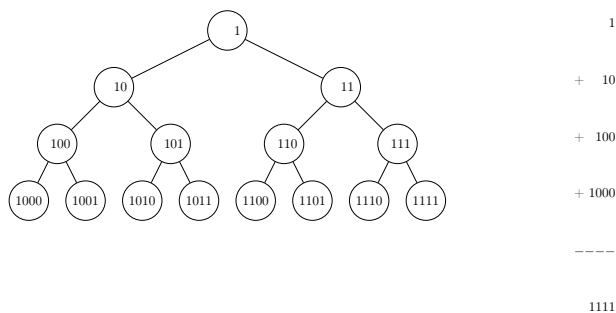


Figure 1: Perfect Binary Tree

3.2.3 Height of a Perfect Binary Tree

A perfect binary tree is defined to be a binary tree in which each non-leaf node has exactly two children. See Figure 1 and notice that the topmost level contains a single (i.e., 2^0) node, while the second level from the top contains two (i.e., 2^1) nodes, etc. Define level k to be the level at distance k from the topmost level, when such a level exists. Then, in a perfect binary tree, level k contains 2^k nodes since the all of the nodes at that level can be indexed by all of the bitstrings of length $k + 1$ that begin with a 1 and there are 2^k bitstrings of this kind. So, notice that when the tree has $h + 1$ levels then the node count equals $1 + 2 + 4 + \dots + 2^h$. Using a radix 2 perspective, this sum can be rewritten as $1 + 10 + 100 + \dots + 1000 \dots 0$ with the last radix 2 number containing h zeros. Notice that, as in Figure 1 the sum is easily seen to be the radix 2 number $1111 \dots 1$ containing $h + 1$ ones since each level contributes exactly a single digit 1. So, the number of nodes is $1111 \dots 1 = 10000 \dots 0 - 1 = 2^{h+1} - 1$ and $h + 1 = \text{len}_2(n)$ so the height of the tree, h , is $O(\text{len}_2(n))$.

4 Conclusion

The results from the last section gives students intuition about why many algorithms have worst case running time $O(\text{len}_2(n))$ when n is the number of nodes and the binary tree involved is perfect or “near” perfect (i.e., *balanced*). In particular, once students understand the connections developed in this paper, students often have an newfound appreciation for the $O(\text{len}_2(n))$ complexity for the binary search tree algorithms *insert*, *delete*, and *search*, as well as for the heap algorithms *siftDown* and *deleteMax*. In addition, students seem to be more willing to play around with binary expressions of the form $\text{len}_2(1) + \text{len}_2(2) + \text{len}_2(3) + \dots + \text{len}_2(n)$ to potentially realize its $O(n \cdot \text{len}_2(n))$

nature than they are to do the same kind of work with the analogous expression involving the binary logarithm, lg . Lastly, almost every student who has learned about the relationship between $log_b(n)$ and $len_b(n)$ immediately appreciates how much smaller these functions are compared to n when n is large. In particular, students instantly know that $log_{10}(1,000,000) \approx 7$ and, therefore, that $log_{10}(1,000,000) < 1,000,000$.

References

- [1] Garvin Brod. “Toward an understanding of when prior knowledge helps or hinders learning”. In: *npj Science of Learning* 6.1 (2021), p. 24. DOI: 10.1038/s41539-021-00103-w. URL: <https://doi.org/10.1038/s41539-021-00103-w>.
- [2] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844.
- [3] Michael Kart. “Making change: rigorous software construction as experiential learning”. In: *J. Comput. Sci. Coll.* 36.7 (Apr. 2021), pp. 76–85. ISSN: 1937-4771.
- [4] Donald E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN: 0201896850.

Learning Parallelism Through an Unplugged Class Activity

Conference Workshop

*Matthew Toups¹, Anurag Dasgupta², Venkat Margapuri³,
Simon Shamoun⁴, Shubbbhi Taneja⁵*

*¹Computer Science Department
Tulane University, New Orleans, LA 70118
mtoups3@tulane.edu*

*²Computer Science and Engineering Technology
Valdosta State University, Valdosta, GA 31698
adasgupta@valdosta.edu*

*³Department of Computing Sciences
Villanova University, Villanova, PA 19085
vmargapu@villanova.edu*

*⁴Computer Science Department
Hofstra University, Hempstead, NY 11549
Simon.Shamoun@hofstra.edu*

*⁵Computer Science Department
Worcester Polytechnic Institute, Worcester MA, 01609
staneja@wpi.edu*

1 Background: PDC in CS education and unplugged classroom activities

Computer Science courses are increasing their coverage of “parallel and distributed computing” (PDC). [4] Adoption of PDC content is growing in many parts of computing education, encouraged by programs such as Center for Parallel and Distributed Computing’s CDER (Curriculum Development and Educational Resources).[3] [5] The rapid development of cloud computing and big data breakthroughs have changed the programming paradigms and con-

tributed to the phenomenal growth of parallel and distributed computing. In traditional sequential programs, the assumption is that execution occurs on only one processor, such that program is sequential in nature. Today, modern computers typically employ multiple processor cores, and are interconnected with high speed networks, leading to the advancement of parallel and distributed computing.

Computer Science instructors generally teach these ideas through computer-based examples and programming-based assignments. The authors intend to enhance those existing assignments and examples by adding a non-computer-based classroom “unplugged activity”[1], to demonstrate the concept of parallelism in computing. The proposed activity the authors conducted specifically involves the topic of parallel computing, in which multiple processors (or threads) divide a problem into sub-problems and then compute solutions simultaneously. This computing problem can be studied in a real-world analog: sorting playing cards by suits and ranks. Tasks are divided among teams of students of varying sizes to explore the benefits and costs of parallel algorithms in this real-world problem. The authors aim to determine the effectiveness of the class activity in helping students learn parallel computing.

2 Workshop Agenda

The authors are faculty at five disparate US-based universities who have collaborated on the topic of PDC unplugged activities. We have conducted IRB-approved research within our classrooms at three campuses so far, with the other two in progress. Our goal is to measure the effectiveness of unplugged activities in our classrooms.

2.1 Introductions

We will begin by an introduction of the presenters and the experiences we bring to this workshop. We would also like the chance to learn about our audience of educators and others: what disciplines do you represent and what student populations do you serve?

2.2 PDC topic introduction

As described in Section 1, PDC is a growing part of CS education. Each of the authors have participated in one or more workshops affiliated with the CDER Center’s NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing. We will address the role this topic plays in our own classrooms.

2.3 “CS Unplugged” activities

“CS Unplugged” is a name for class activities which involve “Computer Science without a computer”. [2] There is a rich collection of such teaching materials available for educators, although more often these deal with topics like algorithms, data formats, or theory of computation. Since unplugged activities offer many well-studied advantages [6], we believe PDC topics will also benefit from further adoption of unplugged classroom activities.

2.4 Case study: sorting playing cards

Our example unplugged activity takes place during one class period, wherein the instructor starts by defining the card-sorting activity and demonstrating how cards should be sorted according to suit and rank. Students take a pre-activity survey to determine their pre-existing ideas about what method they would choose to solve this sorting problem. The instructors want to see how large of a team students think is efficient for solving the problem. Then, the instructor divides the class into teams of different sizes and distributes a shuffled deck of playing cards to each team, who then devises a strategy for sorting the cards. The groups time themselves sorting the deck of 52 playing cards. Afterwards, the class discusses the results and considers this in the context of a parallel computing problem. Finally, the instructors give the students a post-activity survey to measure how their perception about parallel computing has changed based on their hands-on experience with the class activity. In the post survey students describe some advantages and disadvantages of having a larger team vs. a smaller one. Depending on the level of the students, they may be asked to reflect on how their method translates to memory sharing, communication, and control in parallel architectures in higher level courses.

2.5 Research studies on effectiveness

Our group is conducting research on the effectiveness of this classroom intervention. We are doing this in different courses, at different universities with disparate student populations. We will share some preliminary results, and also provide some suggestions for how to conduct research studies in a classroom setting.

2.6 Feedback from attendees

We’d appreciate feedback from the attendees, both about the case study activity, but also about ideas for creating new activities with similar goals.

References

- [1] Tim Bell and Jan Vahrenhold. CS unplugged—how is it used, and does it work? *Adventures between lower bounds and higher altitudes: essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday*, pages 497–521, 2018.
- [2] Suzanne J Matthews. PDCunplugged: A free repository of unplugged parallel distributed computing activities. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 284–291. IEEE, 2020.
- [3] Sushil K Prasad, Almadena Chtchelkanova, Sajal Das, Frank Dehne, Mohamed Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, et al. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: core topics for undergraduates. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 617–618, 2011.
- [4] Rajendra K. Raj, Carol J. Romanowski, John Impagliazzo, Sherif G. Aly, Brett A. Becker, Juan Chen, Sheikh Ghafoor, Nasser Giacaman, Steven I. Gordon, Cruz Izu, Shahram Rahimi, Michael P. Robson, and Neena Thota. High performance computing education: Current challenges and future directions. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '20, page 51–74, New York, NY, USA, 2020. Association for Computing Machinery.
- [5] Erik Saule, Kalpathi Subramanian, and Jamie Payton. We need community effort to achieve pdc adoption! In *2021 IEEE 28th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*, pages 43–49, 2021.
- [6] Michael Weigend, Jiří Vaníček, Zsuzsa Pluhár, and Igor Pesek. Computational thinking education through creative unplugged activities. *Olympiads in Informatics*, 13:171–192, 2019.