

# The Journal of Computing Sciences in Colleges

Papers of the 28th Annual CCSC  
Northeastern Conference

April 12th-13th, 2024  
The College of Saint Rose  
Albany, NY

Baochuan Lu, Editor  
Southwest Baptist University

Jeremiah W. Johnson, Regional Editor  
University of New Hampshire

**Volume 39, Number 8**

**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

|  |           |
|--|-----------|
| <b>The Consortium for Computing Sciences in Colleges Board of Directors</b>  | <b>7</b>  |
| <b>CCSC National Partners</b>  | <b>9</b>  |
| <b>Welcome to the 2024 CCSC Northeastern Conference</b>  | <b>10</b> |
| <b>Regional Committees — 2024 CCSC Northeastern Region</b>   | <b>11</b> |
| <b>Invited Speaker Bios</b>  | <b>13</b> |
| <b>Towards Student Engagement: Drones in Introductory CS Courses</b>   | <b>14</b> |
| <i>Adam Albina, Michael Huelsman, Saint Anselm College</i>   |           |
| <b>Programming Skills as a Gateway to Proof Writing Proficiency</b>  | <b>27</b> |
| <i>Steve Earth, Jeremy Johnson, Bruce Char, Drexel University</i>  |           |
| <b>Agile Practices and their Role in Enhancing Productivity in Student-Run Software Development Projects</b>   | <b>39</b> |
| <i>Sandeep Mitra, Eric Rosenheim, SUNY Brockport</i>   |           |
| <b>GraySim: An OS Scheduling Simulator</b>   | <b>55</b> |
| <i>Sierra Zoe Bennett-Manke, Maria R. Ebling, United States Military Academy</i>   |           |
| <b>A Pleasant Surprise: A Classic Assignment and an Offbeat Assessment for PDC, HPC, and related KU Coverage</b>   | <b>70</b> |
| <i>Nicholas S. Rosasco, Andrew Paxson, Ethan Hawk, Valparaiso University</i>   |           |
| <b>Evaluating the Efficacy of Peer-Created Worked-Example Videos in a Computer Systems Course</b>  | <b>83</b> |
| <i>Grace Kim, Dylan Green, Suzanne J. Matthews, United States Military Academy</i>   |           |
| <b>Enhancing Empathy and Inclusivity in Computer Science Education: An Empirical Study on Accessibility Interventions for Undergraduate Students</b>                 | <b>98</b> |
| <i>Jami Cotler, Dmitry Burshteyn, Amani Walker, Siena College, Eszter Kiss, James Cook University, Megan Hale, Brockport College, John Slyer, SkyHigh Adventures</i> |           |

**Evolving Cybersecurity Education: An Analysis of the GenCyber Teacher Academy’s Progression from 2022 to 2023 and Beyond** 113  
*Elizabeth A. Radday, EdAdvance, Mehdi Mekni, Liberty Page, Ardiana Sula, Laura Brown, University of New Haven*

**Can ChatGPT Pass a CS1 Python Course?** 128  
*James S. Sharpe, Ryan E. Dougherty, Sarah J. Smith, United States Military Academy*

**Infusing Computational Thinking into a Computer Science Gateway Course** 143  
*Younes Benkarroum, Mohammad Q. Azhar, Borough of Manhattan Community College*

**Engaging CS1 Students with Audio Themed Assignments** 158  
*Matthew McQuaigue, Mack Larson, Philip Smith, Sydney Melech, Kalpathi Subramanian, Erik Saule, University of North Carolina at Charlotte*

**Mitigating Use of Artificial Intelligence in Student Assignments** 173  
*Michael Jonas, University of New Hampshire*

**GitKit: Teaching Git and GitHub/GitLab Workflow in an Authentic Context — Workshop** 182  
*Grant Braught, Dickinson College, Stoney Jackson, Western New England University, Karl R. Wurst, Worcester State University*

**Guiding Students to Discover CS Concepts and Develop Process Skills Using POGIL — Workshop** 184  
*Olga Glebova, University of Connecticut, Ali Al-Farris, Worcester State University, Nanette Vielleux, Simmons University*

**Bridging the Empathy Gap: Hands-On Exploration of Human-Empathy Accessibility Learning (HEAL) Interventions in Computer Science Education — Workshop** 186  
*Maggie Frechette, Annabel Fisher, Mohammad Shaikh, Jon LeFrois, Dmitry Burshteyn, Jami Cotler, Siena College, Sam Malachowsky, Daniel Krutz\*, RIT*

**Designing Technical Presentation with Good Narrative Structure — Workshop** 191  
*Paul Kim, Bridgewater State University*

**Fundamental Cybersecurity Hands-on Exercises using an in-house Cyber Range — Workshop** **193**  
*Giuseppe Sena, Ryan Fried, Shamsi Moussavi, Massachusetts Bay Community College*

**Navigating Feasibility: Choosing Service-Learning Projects for Academic Fit — Conference Tutorial** **196**  
*Stan Kurkovsky, Chad Williams, Central Connecticut State University, Mikey Goldweber, Denison College, Nathan Sommer, Xavier University*

**Incorporating Computing for Social Good in Education — Conference Tutorial** **198**  
*Heidi J. Ellis, Western New England University, Gregory W. Hislop, Drexel University*

**Assignment Management and Automated Assessment Using Git-keeper — Conference Tutorial** **200**  
*Ben Coleman, Moravian University, Nathan Sommer, Xavier University*

**Using Continuous Integration (CI) For Managing Programming Assignments — Conference Tutorial** **202**  
*Sunjae Park, Wentworth Institute of Technology*

**A Process for Distinctive Curricular Design for Liberal Arts Computing Programs — Conference Tutorial** **204**  
*Jakob Barnard, University of Jamestown, Grant Braught, Dickinson College, Janet Davis, Whitman College, Amanda Holland-Minkley, Washington & Jefferson College, David Reed, Creighton University, Karl Schmitt, Trinity Christian College, Andrea Tartaro, Furman University, James Teresco, Siena College*

**Using the Accessible Learning Labs for Accessibility and Machine Learning Education — Conference Tutorial** **207**  
*Carla Lopez, Daniel E. Krutz, Kyle Messerle, Samuel Malachowsky, Rochester Institute of Technology, Jami Cotler, Siena College*

**Assessing Risks, Challenges and Opportunities of Generative AI in Computer Programming Education — Lightning Talk** **210**  
*Sa Liu, Interactive Media, Brian Grey, Chad Chu, Phillip Grim, Thomas McManus, Harrisburg University, Ryan Watkins, George Washington University*

|  |            |
|--|------------|
| <b>Unambiguous Non-Functional Requirements Specification</b>   |            |
| – <b>An Extended Use-Case — Lightning Talk</b>   | <b>212</b> |
| <i>Unnati Shah, Utica University, Krupa Patel, Bhagwan Mahavir University</i>  |            |
| <b>Redesigning Introduction to Programming Course Work using a Project-based Learning Approach — Lightning Talk</b>  | <b>214</b> |
| <i>Ali Al-Faris, Saba Kadady, Worcester State University</i>   |            |
| <b>Exercises on Distributed List Operations — Lightning Talk</b>   | <b>216</b> |
| <i>Jan Plaza, SUNY Plattsburgh</i>   |            |
| <b>An Accelerated Java Programming Summer Program</b>  |            |
| — <b>Poster Abstract</b>   | <b>218</b> |
| <i>Ching-yu Huang, Paolien Wang, Kean University</i>   |            |
| <b>Code Samurai Program: Empowering Students in Computer Science and Technology — Poster Abstract</b>                | <b>220</b> |
| <i>Paolien Wang, Ching-yu Huang, Patricia Morreale, Kean University</i>  |            |
| <b>Tech Startup: A Multi-Level Multi-Semester Project Course</b>   |            |
| — <b>Poster Abstract</b>   | <b>222</b> |
| <i>Kevin McCullen, SUNY at Plattsburgh</i>   |            |
| <b>Beyond the Pixels: An Interdisciplinary Approach to Image Analysis in Astronomy — Poster Abstract</b>             | <b>224</b> |
| <i>Nicole Granucci, Chetan Jaiswal, Quinnipiac University</i>  |            |
| <b>Redesigning Introduction to Programming Course Work using a Project-based Learning Approach — Poster Abstract</b> | <b>228</b> |
| <i>Ali Al-Faris, Saba Kadady, Worcester State University</i>   |            |
| <b>Automating Paper-based Peer Reviews — Poster Abstract</b>   | <b>230</b> |
| <i>Delbert Hart, SUNY Plattsburgh</i>  |            |
| <b>Reviewers — 2024 CCSC Northeastern Conference</b>   | <b>232</b> |

## 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 Northeastern Conference

Welcome to Albany, New York, and the College of Saint Rose, for the Twenty-eighth Annual Consortium for Computing Sciences in Colleges Northeast Region Conference.

Our program features two distinguished invited speakers, Stephen MacNeil of Temple University and Shaghayegh Sahebi of SUNY Albany. In addition, we have a broad range of topics covered by paper presentations, lightning talks, workshops, tutorials, and faculty and student research posters. On Friday morning, we are hosting the traditional programming contest. On Friday afternoon, we include some student-focused sessions. In addition to the career fair, there will be a session where programming contest participants and organizers can discuss the problems.

This conference would not happen without the efforts of the outstanding conference committee and dedicated board. The success of this conference is the result of the hard work of those groups, plus the reviewers, session chairs, and countless other volunteers. The conference continues to be selective; we accepted 13 of 27 papers for an acceptance rate just over 48%. This ensures the high-quality program that everyone expects.

We also want to thank the National Partners whose support made the conference possible: acm2y and Rephactor. Additional support is provided to the conference for being in cooperation with the ACM Special Interest Group on Computer Science Education (SIGCSE).

We are pleased to be hosting this year's conference for the second time at The College of Saint Rose, on its beautiful urban campus in Albany's vibrant Pine Hills neighborhood. As many likely are aware, this is the last academic year in the 100+ year history of Saint Rose. We are grateful to everyone at the College who have played important roles in the planning and execution of the conference, even as they are dealing with the challenges of Saint Rose's final year.

Ian MacDonald  
The College of Saint Rose  
Jim Teresco  
Siena College  
Conference Co-Chairs



## 2024 CCSC Northeastern Conference Steering Committee

|  |       |                                      |
|--|-------|--------------------------------------|
| Ian MacDonald, Conference Co-chair                         | ..... | The College of Saint Rose            |
| Jim Teresco, Conference Co-chair                           | ..... | Siena College                        |
| Mark Bailey, Program Co-chair                              | ..... | Hamilton College                     |
| Yana Kortsarts, Papers Co-chair                            | ..... | Widener University                   |
| Chetan Jaiswal, Papers Co-chair                            | ..... | Quinnipiac University                |
| Ting Liu, Panels/Tutorials/Workshops Co-chair              | ..... | Siena College                        |
| Jordan Crouser, Panels/Tutorials/Workshops Co-chair        | ..... | Siena College                        |
| Dan Rogers, Faculty Posters and Lightning Talks Co-chair   | ..... | SUNY Brockport                       |
| Joan DeBello, Faculty Posters and Lightning Talks Co-chair | ..... | St. John's University                |
| Sandeep Mitra, Undergraduate Posters Co-chair              | ..... | SUNY Brockport                       |
| Adita Kulkarni, Undergraduate Posters Co-chair             | ..... | SUNY Brockport                       |
| Liberty Page, Undergraduate Posters Co-chair               | ..... | University of New Haven              |
| Drew van der Poel, Undergraduate Posters Co-chair          | ..... | Northeastern University              |
| Karyn Doke, Undergraduate Posters Co-chair                 | ..... | The College of Saint Rose            |
| Ali Erkan, Faculty Cohorts Co-chair                        | ..... | Ithaca College                       |
| Mike Gousie, Faculty Cohorts Co-chair                      | ..... | Wheaton College                      |
| Larry D'Antonio, Speakers                                  | ..... | Ramapo College of New Jersey         |
| Rick Kline, Registration Co-chair                          | ..... | Pace University                      |
| Garrett Dancik, Registration Co-chair                      | ..... | Eastern Connecticut State University |
| Del Hart, Programming Contest                              | ..... | SUNY Plattsburgh                     |
| Paul Olsen, Programming Contest                            | ..... | The College of St. Rose              |
| Darren Lim, Programming Contest Co-chair                   | ..... | Siena College                        |
| Mark Gilder, Programming Contest Co-chair                  | ..... | The College of Saint Rose            |
| Sharon Jaramillo, Local Arrangements                       | ..... | The College of Saint Rose            |
| Kevin McCullen, Vendors and Partners Co-chair              | ..... | SUNY Plattsburgh                     |
| Ingrid Russell, Vendors and Partners Co-chair              | ..... | University of Hartford               |
| Ali Erkan, Previous Conference Chair                       | ..... | Ithaca College                       |
| Adita Kulkarni, Incoming Conference Chair                  | ..... | SUNY Brockport                       |

### Regional Board — 2024 CCSC Northeastern Region

|  |       |   |
|--|-------|---|
| James Teresco, Regional Representative | ..... | Siena College                             |
| Jeremiah Johnson, Editor               | ..... | University of New Hampshire at Manchester |
| Richard Kline, Registrar               | ..... | Pace University                           |
| Sandeep Mitra, Secretary               | ..... | The College at Brockport                  |
| Adrian Ionescu, Treasurer              | ..... | Wagner College                            |
| Delbert Hart, Webmaster                | ..... | SUNY Plattsburgh                          |



## Invited Speaker Bios

**Stephen MacNeil** is an Assistant Professor in the Department of Computer and Information Sciences at Temple University where he directs the Human-Computer Interaction Lab. His work in computing education spans inclusive classroom design, scalable undergraduate research ecosystems, and generative AI in education. His work has been published in top ACM venues including SIGCSE, ICER, ITiCSE, CHI, UIST, and CSCW. Committed to fostering undergraduate research, over 50 undergraduate students have published a paper or poster during their time in the lab.

**Shaghayegh Sahebi** is an Associate Professor of Computer Science at the University at Albany – SUNY and the director of Personalized AI (PersAI) Lab. She received her Ph.D. and M.Sc. in Intelligent Systems Program from the University of Pittsburgh. Also, she has an M.Sc. and B.Sc. in Computer Science/Engineering from the University of Tehran and Sharif University of Technology, respectively. Her broad research interests include machine learning and data mining for human-centered applications, such as educational, recommendation, and health application systems. Her research has been supported by three NSF awards, including a CAREER award in 2021 and a CRII award in 2018. Her research has been featured in peer-reviewed publications in major venues such as The International Educational Data Mining (EDM), ACM Web conference, ACM Recommender Systems (RecSys), The AAAI Conference on Artificial Intelligence (AAAI), and Artificial Intelligence in Education (AIED). She has also been an active member of the research community, e.g., serving as the program co-chair of the Fourteenth International Conference on Educational Data Mining (EDM 2021).

# Towards Student Engagement: Drones in Introductory CS Courses\*

Adam Albina and Michael Huelsman  
Department of Computer Science  
Saint Anselm College  
100 Saint Anselm Drive  
Manchester, NH 03102  
{aalbina, mhuelsman}@anselm.edu

## Abstract

Student engagement has been shown to increase academic achievement. We apply learning engagement strategies in introductory computer science courses using unmanned aerial vehicles (UAVs), commonly referred to as drones. Using programming to control and fly drones reinforces the basic concepts of programming in an interesting and enjoyable way for first year programming students in a CS1 course. Additionally, we apply similar but less technical engagement strategies in a general education course with no prerequisites, and engage students with issues in modern technology through the medium of drones. Many in higher education see the introduction of complex topics as an important method of engaging students and therefore of improving learning outcomes. This leads many to offer courses in areas like robotics, AI, and game development. A potential problem with this approach is the level of knowledge required for these courses precludes their use as an engagement strategy in introductory coursework where they might attract the attention of a broader set of students. To this end we looked into adding drone technology to introductory and non-major CS courses. While preliminary, these results suggest a high level of student interest in the topic and promising engagement from both CS major and non-CS-major students.

---

\*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

Student engagement has been shown to have a positive impact on academic achievement [8], which holds true specifically in introductory computer programming courses employing learning engagement strategies [9, 3, 4, 10]. Finding engaging programming problems in introductory Computer Science (CS) courses for first-year students across multiple disciplines has proved a challenge. Course topics and project selections are often made with an eye towards technologies that are at the forefront of the news or general student interest [1, 5]. Examples include game programming, robotics, 3D printing, etc. Project-rich problems in these areas require knowledge from the areas of Algorithmic Foundations (AF), Foundations of Programming Languages (FPL), and Software Development Fundamentals (SDF), as outlined in the gamma version of the Computer Science Curricula 2023 [2]. This knowledge is likely lacking in first-year students. Additionally, at many *Liberal Arts* institutions, Computer Science departments not only support their programs but provide knowledge transfer across many majors in support of the institutional mission [6]. Such dual-service environments amplify the problems caused by requiring high levels of pre-requisite coursework. Relegating motivational laboratory and classroom learning engagement strategies to upper-level classes misses the mark for us on two fronts. First, we seek to motivate and excite first-year computer science students to increase their academic success as well as keep them in the major. Second, we aspire to provide courses that serve both the major and the undeclared major as they take introductory courses in search of their interest. At many Liberal Arts institutions, a large number of students enter the institution as undeclared with respect to major, making this type of successful exploration critical in the decision process. Finally, CS programs at Liberal Arts institutions are generally supported by three to four faculty members [6] which severely limits the number of courses that can be offered in a term.

This work provides insights into the integration of a learning engagement strategy using programmable quad-copters, or drones, in an introductory computer science course over a period of two years. Autonomous flight, object detection and response using computer vision, and flight emergency exception handling are covered after students have learned fundamental software development in the course. The success of the engagement strategy in the programming course and the desire to address the Society, Ethics, and Professionalism (SEP) knowledge area [2], with respect to Unmanned Aerial Vehicles (UAV), prompted the creation of an interdisciplinary course on *Drones and Society*, which was open to all majors.

## 2 Background

Our first experience with the development of a drone-related curriculum came in the form of a week-long residential summer camp. The camp program included both small-form, programmable drones and larger pilot-controlled kit drones. While we were charged with overall camp administration, our primary pedagogical role was teaching basic programming and extending those concepts to the smaller programmable drones. To the extent possible, the week was divided equally between the programmable drones and the larger kit drones. This allowed participants to understand the engineering principles behind building drones, the aerospace forces at play, as well as what drones are capable of with programming, once they reach a sufficient level of sophistication. The catalyst for the drone camp was a partnership with the Aerospace Robotics Competition run by the non-profit STEM-ED, the Saint Anselm College Center for Ethics, and the Department of Computer Science at Saint Anselm College.

While the camp proved successful in the primary goal of promoting STEM, we also found a significant secondary effect. Given the difficulties in managing a group of 12-15 high school-age young adults, we required several camp counselors, primarily made up of CS majors from the department. In order to ensure that the counselors were effective in reducing the overall teaching load and able to assist camp participants, each counselor was trained in the API of the programmable drones. Given that the students were already CS majors who had taken an introductory course in programming (Python), we were able to onboard them quickly and they were effective assistants during programming instruction and challenges. The enthusiasm with which our student counselors engaged with the programmable drones demonstrated a marked interest in drone programming and general drone technology. This background provided us with the foundations of what we used in our endeavors to introduce drones in introductory computer science coursework.

## 3 Drone Curriculum

This section is split between two courses: *Computing I* (CS1) and *Drones in Society*. These courses represent the breadth of pertinent courses where we have implemented drone curricula. While some elements are shared between the two courses, their audiences are somewhat distinct. CS1 is primarily taken by students with a direct interest in programming and CS, while *Drones in Society* was meant for a broader audience of students who may have no interest in programming. Despite both courses being geared for different audiences, many of the curricular techniques used are shared between them.

### 3.1 CS1 Course

In determining our approach to introducing drone programming in our first programming course sequence, we focused on three simple objectives. The first was to allow the students a level of experimentation and hands-on work with drone APIs and programmable drone hardware. This pedagogical technique seeks to foster a sense of discovery without direct instruction or solutions. The second was to allow for sufficiently advanced activities in computer vision and object detection without requiring students to understand these machine learning techniques completely. The third was to use the drone work as reinforcement to previously covered introductory programming concepts. We settled on the following learning outcomes for this portion of the course in addition to the learning outcomes already in place for CS1:

1. Develop programs that use APIs to access drone data (telemetry and video).
2. Develop programs that reinforce fundamental programming constructs to control drone movement safely.
3. Develop programs that use artifacts retrieved from drones to perform specific tasks (classification, tracking).

This CS1 course is currently conducted as a flipped classroom environment following [7, 11] with approximately 60 micro-lectures online, so students are accustomed to preparing for class, taking a quiz on the video prior to class, and participating in exercises during class time.

We investigated several drones for educational use and found only two that would allow for direct programming using Python. We chose the DJI Ryze Tello platform due to its compact size (allowing for use indoors), accessible API, forward-facing 720p camera, and low cost. This product has no GPS capability and is not intended for outdoor use. DJI provides a Python API in the `djitellopy` package downloadable from GitHub or installable with package management software. The platform also allows for UDP socket connections to the drone through which commands can be transmitted directly. The `DJITelloPy` package also installs the `openCV` Python and `numpy` packages. A client computer connects to the drone using a WiFi access point, which the drone broadcasts. For this course, we used the `djitellopy` package as it simplifies the act of connecting and communicating with the drone. This simplification is needed because most students, in a CS1 course, have no background in networking and don't understand sockets. In particular, this portion of the curriculum covers approximately 2 weeks toward the end of the CS1 course. Students have already completed coursework, assessments, and hands-on labs that cover standard CS1 programming topics such as variables, data types,

flow control, functions, object-oriented constructs, and graphics programming in the Python language. At the point when drones are introduced, students are fairly well versed in problem-solving, program design, and logic. The typical class size for our CS1 course is 18-20 students.



Figure 1: Aruco Marker and Tello drone in flight.

We cover the topic in three modules. The first module introduces students to connecting to the drone and programming the basic movements of the drone (forward, backward, left, right, altitude, and rotation.) For this work, students form teams with each team member being required to demonstrate their code during class/lab. To aid with group work, the classroom we use contains group collaboration technology in the form of a pod configuration with TVs to which students can connect their laptops and project their display. Additionally, this classroom has sufficient free, uncluttered space for flight activities. Typical problems students have to solve during class time and lab are shown in Table 1. These tasks are evaluated by instructor observation.

| Task  | Learning Outcome       |
|---|------------------------|
| Drone takeoff,<br>display battery power,<br>land drone                              | Control drone movement |
| Drone take off,<br>increase altitude by 110 cm,<br>fly an 80cm square               |                        |
| Drone take off,<br>increase altitude by 110 cm,<br>fly an 80cm equilateral triangle |                        |

Table 1: Module 1 typical tasks



The second module reinforces the first and adds interactive flight using keystrokes and includes taking pictures with the onboard camera and saving them to the program execution directory, as well as emergency flight termination. Students are encouraged to explore the API during this module and create programs that use keyboard input to execute commands in the API. Students typically add flying curves, flipping the drone, and sequenced movement (forward 40cm and left 40cm, etc.) to controls they have already programmed. Typical problems students have to solve during class and lab time are shown in Table 2.

| Task  | Learning Outcome                             |
|---|--|
| Key press to take off,<br>key press for all movements,<br>key press for picture | Control drone movement,<br>access drone data |
| Key press to flip forward,<br>key press to fly 40cm circle                      | Control drone movement                       |
| Build movement GUI,<br>button click to fly 40cm circle                          | Control drone movement,<br>access drone data |

Table 2: Module 2 typical tasks



Figure 2: Student programmed flight of a Tello drone with ArUco marker.

The third module builds on the first two and adds a computer vision component with openCV frame markup, ArUco marker detection, and ArUco marker movement tracking. Obviously, CS1 students will likely not have the necessary

background to deal with numpy image arrays and openCV complexities. We provide an instructor-developed ArUco detector Python class for this reason. The Python class provides methods to process a single video frame, draw a red box around detected markers, and draw the ArUco numbers in the box. The detector will detect all markers in the frame, but will only store the upper left and bottom right points (opposing corners) of the framing box of the marker along with the marker number of the last marker detected. When the next frame is processed, the attributes of the last marker are cleared or replaced with the new last marker attributes. This provides enough information for students to calculate the center of the last marker in pixels and adjust the rotation of the drone to track marker movement in the 1080 x 720 image field. This results in students building programs that turn on the video stream of the drone, search for a marker, and process frames from the live video. They instantiate a detector object and invoke the appropriate methods to detect the marker and retrieve its artifacts (classification). After successful detection, the student must take in-flight action to move the drone appropriately in response to a moving ArUco marker (tracking). These exercises are completely autonomous with no manual intervention besides emergency shutdown allowed. Typical problems students have to solve during class time and lab are shown in Table 3.

| <b>Task</b>  | <b>Learning Outcome</b>   |
|--|---|
| Take off,<br>display battery power                                   | Control drone movement  |
| Implement a search pattern   | Control drone movement  |
| Read video frame,<br>process frame for marker,<br>display live video | Control drone movement,<br>access drone data,<br>perform classification |
| Calculate center of marker,<br>adjust drone to track marker          | Control drone movement,<br>access drone data,<br>perform tracking       |

Table 3: Module 3 typical tasks

Students are required to accomplish many of the tasks in the classroom during normal instruction. However, we also include a graded lab in which students have to develop and demonstrate their code live for the lab instructor. This lab is the last of 12 labs over the course of the semester and, like all of our labs, includes a lab manual that contains the problems to be solved during the

lab or prior to the next lab. We ensured that the drone problems were solvable during lab time, as we do not allow the students to take the drones with them.

### 3.2 Interdisciplinary Course

As is common in Liberal Arts colleges, our institution contains an expansive set of general education requirements (referred to as the *core*.) Given that all students must complete the core in order to graduate there is a significant internal market for courses which fill core requirements while also being open to the general population of students. Given the importance of understanding modern technology and our previous drone-based curricular development we moved to construct a course which would fulfill a core requirement, have no prerequisites, and engage students with issues in modern technology through the medium of drones. This course, which we named *Drones in Society*, takes some of the software and instruction from our CS1 curriculum and adapts it for use in a setting where students are not expected to know, or learn, programming or software development. In addition, this course was designed to fulfill one of a student's *Writing Intensive* core requirement courses, thus alongside instruction relevant to drones, the students needed to produce a large amount of written artifacts, including a final research paper.

The resulting curriculum for *Drones in Society* saw the course split between three different types of lecture: writing, lightly technical drone instruction, and a seminar-style discussion. The writing lectures focused solely on the process of writing long-form non-fiction, similar to any English composition course. Discussion lectures were focused on material read outside of class on drones, and drone-related topics, with a particular focus being placed on drone usage and ethics. Finally, the technical drone instruction provided an overview of drone technology and how it worked. Due to the course's purpose as a core course (and not a course for only those interested in becoming CS majors) the technical instruction provided was primarily overview, with few topics being investigated more intensely than surface level. The technical instruction was designed to help students understand the context and problems in drone technology so that they could better understand readings and intelligently talk about modern drone technology, without overreaction or misunderstanding.

Hands-on activity-based learning was also integrated into the course. These "Flight Days" saw students use the Tello drones (in both remote control and programmatic contexts) to complete a singular task such as navigation, facial recognition, and environmental sensing. Since we could not rely on students having any foreknowledge of programming the amount of scaffolding was increased over that used in CS1. While the djitellopy module contained a great deal of the functionality we needed for this scaffold, the amount of additional features and concerns with regards to performance meant that it was prudent

for use to create our own Python module for use with the class. This module allows not only for programmatic control of the drone, but also for ArUco marker detection, RC control, a heads-up display (HUD) during flight, and face recognition. Additionally, since we control the code we can add or remove features as needed and address bugs/glitches in a timely fashion. This also mean that students need only install a single package (and it's dependencies) and they are ready to interface with the drones.

## 4 Results and Analysis

In an effort to determine if adding drone programming to the CS1 curriculum was beneficial, we looked at four years of past data. The first two years did not include the drone curriculum and the second two years included the drone curriculum. We analyzed enrollment data to determine how many declared students taking CS1 progressed to CS2 in the next term or in subsequent terms. Our hypothesis was that adding the engaged learning of drone programming to CS1 contributes to students enrolling in CS2. However, our progression from CS1 to CS2 in the first two years of analysis was 100% for CS-related majors. Naturally, we abandoned the method as not yielding useful information. We analyzed major changes for a first or second major subsequent to taking the CS1 course for both groups. We hypothesized that students changed their major after taking CS1 at a greater rate during the drone programming years than the previous two years. Analysis revealed that students changed or added a second CS major more frequently in the years drone programming was included in CS1, see Table 4. This suggested that it is possible that drone programming was a factor, certainly among others, for adding the CS major. This warrants further research for confounding factors in the outcome.

| Year      | Major Changes |
|-----------|---------------|
| 2020-2021 | 3             |
| 2022-2023 | 8             |

Table 4: Major Changes to CS Majors

Finally, we also added four questions to the course evaluation for the most recent year, which focused on the drone programming in CS1. These questions allowed us to solicit feedback from students. We provided the following four statements for agreement on a six point Likert scale:

1. Learning how to control/fly drones with Python was a good use of what I learned in this course.

2. The drone work we did in class was useful in reinforcing programming concepts.
3. I enjoyed controlling/flying drones in this course.
4. The drone programming portion of this course should continue to be included.

Overwhelmingly, the responses were very positive for all four evaluation statements and there were no responses in the Disagree or Strongly Disagree categories. These categories are omitted from Figure 3. Response rate for the course evaluation was 72%.

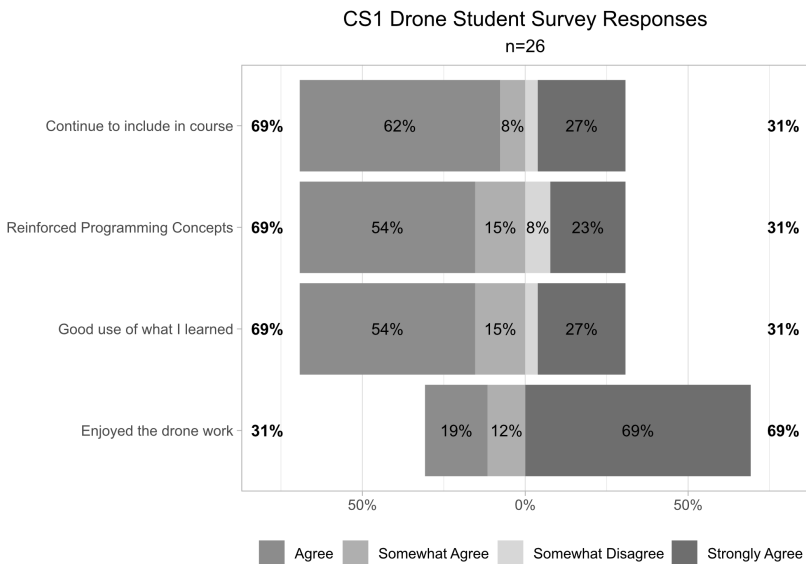


Figure 3: Divergent Visualization of Likert Scale Evaluations

We believe that including engaged learning using drones in our CS1 course has resulted in increased interest from students. While the analysis does not support making claims of improved progression from CS1 to CS2, it does suggest that including engaged learning with drones could be a factor in students declaring the major after taking the course. More research is required before any broad conclusions can be made.

The results generated by our Drones in Society course are less concrete, but still positive. During the first offering of Drones in Society we saw an

enrollment where the majority of students were non-majors and non-minors. The students represented many non-CS fields including politics, environmental science, business, and criminal justice. The enrollment for our second offering is currently similar in terms of academic diversity. We attribute this diverse enrollment to student interest in drones and the course's inclusion in the core. While completion of the core was likely a component of student reasoning for taking the course many other disciplines offer courses which would complete the same core requirement, thus we feel safe in saying that student interest did take a central role in student decision making. This likely demonstrates that students have a desire to learn about technical topics, particularly those that impact their world, in a minimally technical manner.

Internally, the course benefited from having academically diverse students due to the seminar nature of some lectures. Anecdotally, the CS students in the course tended to focus primarily on technical topics while students from other majors focused on drone application in their fields. For example, one environmental sciences major wrote their final paper over the use of drones for remote sensing, particularly for detection of signs of forest health. This meant that each discussion contained both low level and high level considerations for the use of drone technology in the modern world.

The results of the Drones in Society course are two fold. First, we saw students who would otherwise not take a CS course engaging with CS material and second, the course allowed us to continue developing curricula and strategies for introducing complex technical topics in introductory coursework. As it stands the course exists primarily to engage the wider student population in pertinent technical topics, while allowing CS students to complete a core requirement in their field.

One drawback of Drones in Society is that as a writing intensive course, it mainly appeals to students in their 3<sup>rd</sup> or 4<sup>th</sup> years, which limits our ability to use it as an effective recruiting tool for CS majors and minors. That being said, our first offering saw a student elect to take our CS1 course the following semester and that student also began pursuing a minor in a CS-related field. This suggests that such service courses are a potential avenue to get students interested in a potential major change, or addition of a minor.

## 5 Conclusions

Our work suggest that not only is it possible to introduce complex CS topics into introductory coursework, but that it may be beneficial to both students and the major overall. It is important to note that while we maintain a hopeful attitude towards the benefits of adding drones to our introductory curricula, this is not an exhaustive study of the topic and represents a limited case study.

We do, however, believe that the data we have collected and our experience with implementing these curricular changes shows the potential of adding modern, complex, and engaging topics to introductory coursework, increasing student engagement, and attracting students to the CS major.

From a more qualitative observation of student engagement in CS1, it was clear that the drone work in class generated significant enthusiasm. Although the in-class/lab programming tasks required the use of logical operations, flow control, and other basic programming constructs, students seemed unaware that they were reinforcing these concepts as they were hurriedly writing programs to make the drone accomplish the given tasks. We do use game development as a final project in the CS1 course but students appeared less engaged with the game development when compared to drones.

While future research is required to solidify the correlation between student engagement, and major enrollment/retention we believe this work opens other possible avenues of research. Longer studies over multiple years may provide further insight into whether increased engagement and interest in CS, started early, will improve student performance, retention in the major, and adoption of the major.

While there is a lot that remains to be seen, we have shown that there is currently student interest in drone technology, and early student introduction to complex topics in CS may help to drive engagement and increase interest in the field of computer science.

## **Acknowledgments**

Instrumental in our drone work was the Aerospace Robotics Competition run by the non-profit STEM-ED, the Saint Anselm College Center for Ethics, and generous donations by Saint Anselm donors.

## References

- [1] Barry Burd et al. “The Internet of Things in Undergraduate Computer and Information Science Education: Exploring Curricula and Pedagogy”. In: *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE 2018 Companion. Larnaca, Cyprus: Association for Computing Machinery, 2018, pp. 200–216. ISBN: 9781450362238. DOI: 10.1145/3293881.3295784. URL: <https://doi.org/10.1145/3293881.3295784>.
- [2] The Joint Task for on Computing Curricula. *Computer Science Curricula 2023 Association for Computing Machinery (ACM) IEEE-Computer Society (IEEE-CS) Association for Advancement of Artificial Intelligence (AAAI)*. Tech. rep. Association of Computing Machinery, 2023.
- [3] Scott Freeman et al. “Active learning increases student performance in science, engineering, and mathematics”. In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415.
- [4] Benjamin Gan Kok Siew et al. “Active Learning Approaches in Information Technology (IT) Pedagogy”. In: *Proceedings of the 52nd ACM Conference on Computers and People Research*. SIGSIM-CPR ’14. Singapore, Singapore: Association for Computing Machinery, 2014, pp. 113–117. ISBN: 9781450326254. DOI: 10.1145/2599990.2600016. URL: <https://doi.org/10.1145/2599990.2600016>.
- [5] Tom Goulding and Rita DiTrolio. “Complex Game Development by Freshman Computer Science Majors”. In: *SIGCSE Bull.* 39.4 (Dec. 2007), pp. 92–99. ISSN: 0097-8418. DOI: 10.1145/1345375.1345423. URL: <https://doi.org/10.1145/1345375.1345423>.
- [6] Amanda Holland-Minkley et al. *Computer Science Curriculum Guidelines: A New Liberal Arts Perspective*. Tech. rep. 2023.
- [7] Celine Latulipe, Audrey Rorrer, and Bruce Long. “Longitudinal data on flipped class effects on performance in CS1 and retention after CS1”. In: *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. Vol. 01. Association for Computing Machinery, Inc, Feb. 2018, pp. 411–416. ISBN: 9781450351034. DOI: 10.1145/3159450.3159518.
- [8] Cui Y Lei H and Zhou W. “Relationships between student engagement and academic achievement: A meta-analysis.” In: *Social Behavior and Personality* 46.3 (2018), pp. 517–528. DOI: <https://doi.org/10.2224/sbp.7054>.
- [9] Vijayalakshmi Ramasamy et al. “Meta-analysis to study the impact of learning engagement strategies in introductory computer programming courses: A multi-institutional study.” In: New York, NY, USA: Association for Computing Machinery, 2022, pp. 40–46. ISBN: 978-1-4503-8068-3. DOI: <https://doi.org/10.1145/3409334.3452060>.
- [10] Kate Sanders et al. “Folk Pedagogy: Nobody Doesn’t Like Active Learning”. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER ’17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 145–154. ISBN: 9781450349680. DOI: 10.1145/3105726.3106192. URL: <https://doi.org/10.1145/3105726.3106192>.
- [11] Gina Sprint and Erik Fox. “Improving student study choices in cs1 with gamification and flipped classrooms”. In: *SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (2020), pp. 773–779. DOI: 10.1145/3328778.3366888.



# Programming Skills as a Gateway to Proof Writing Proficiency\*

Steve Earth, Jeremy Johnson, Bruce Char  
Drexel University  
Philadelphia PA 19104  
`se435@drexel.edu`

## Abstract

The intersection of programming and proof writing skills in computer science education is a relatively unexplored area. This paper presents the beginning of a longitudinal study that explores this intersection by analyzing student programmers' solutions to logic puzzles at the beginning and end of an intermediate computer science (CS) course. These puzzles, requiring skills akin to proof writing but without the need for advanced mathematical knowledge, serve as a tool to evaluate the development of proof writing skills. We examine the correlation between students' puzzle-solving capabilities and their academic performance in the course, controlling for other variables such as prior mathematics courses and GPA. This study aims to bridge the gap in understanding how programming education contributes to the development of proof writing abilities.

## 1 Introduction

### 1.1 Background Context

The ACM/IEEE joint task force on Computing Curricula 2013 [1] states “an ability to create and understand a proof—either a formal symbolic proof or a less formal but still mathematically rigorous argument—is important in virtually every area of computer science”. A 2009 survey done by the Association

---

\*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.

of Symbol Logic [9], showed that only 15% of computer science departments have a separate course for symbolic logic. So, the presumption is that they are getting these skills piecemeal as aspects get incorporated into a variety of their courses. But are we doing an effective job of transferring those proof writing skills to our computer science students? Friske [3] noted that "there is a close analogy between the thought process used in computer programming and those required for writing proofs". Specifically he categorized the overlapping skills to include problem specification, logical design, and symbolic communication. Despite this outline, in 2016, Scherer [8] bemoaned the lack over the past few decades of much empirical research measuring transfer effects to or from these computer science skills and called out to the community to produce measurable results. In the evolving landscape of computer science education, understanding the transfer of skills between programming and proof writing is crucial. This knowledge can inform curriculum design, ensuring that students are equipped with comprehensive analytical and logical skills necessary for the field.

## 1.2 Purpose of Study

Reflecting the points above, Epp [2] notes that "because computer science students also need to learn to operate in a mathematically sophisticated environment, one of the goals ... is to enhance students' logical reasoning and proof-writing abilities". Building on these insights, we aim to explore how formal proof methods taught in computer science classes contribute to developing students' ability to construct solid informal arguments, a skill vital in both academic and practical computing contexts. Toward this end, Izu [4] discusses Perkin's notion of learning transfer in the context of computer science, with the focus of "near transfer" from one programming task to another. In this paper, we extend that idea to "far transfer" and consider the following questions: While ostensibly a computer science class is teaching formal proof methods, is it succeeding in its true greater purpose of teaching students how to make solid informal arguments? And secondly, is this truly an effect of computer courses per se, or could proof writing ability simply be a matter of general academic performance or another related proof course (such as a math class)? The purpose of our study is to grant some insight into answering those questions.

To begin, we must first understand what it is that the prover is beholden to do. Mejía [5] argues the core skills involved in writing a proof are: [i] Exploration of a problem, [ii] Estimating truth of conjectures, [iii] Justification of truth of statements and [iv] Effectively communicating the knowledge in such a way to convince the reader of the validity of the argument. Some texts (notably [7] or Polya's seminal work [6]) make a point of training students to write proofs by

breaking down these skills into various proof templates. Our study emulated proof-writing by carefully choosing a logic puzzle that would emulate each of those skills. (It may be worth noting an initial flawed incarnation of the study naively used Wordle/Mastermind style questions that did not hit all these critical marks). The Mejía criteria (and the supporting texts of math educators) show that logic puzzles designed for this study are a very reasonable proxy for proof writing (see 2.2 for a more detailed breakdown).

## 2 Methodology

### 2.1 Participants

This paper describes the first phase of the study which took place during the summer quarter 2021 with 43 students at Drexel’s Mathematical Foundations of Computer Science course. (The study continued through the Fall where the sections numbers included an additional 125 students; the same results were indicated and will be published in upcoming publications). Virtually all students have had our university’s prerequisite course “Intro to Programming” (the four exceptions were due to testing out or transferring credit from another institution). Similarly, all but two students have taken Calculus 1, which is a requirement for the CS major. The vast majority, 74%, of the class was juniors (the remaining distribution was 2% freshmen, 14% sophomore, and 10% senior). The vast majority, 91%, were computer science majors of some flavor (possibly software engineering, or data science) but we also had 2 Biology majors, 1 Math major, and 1 in Electrical engineering. The course itself uses a recursive programming vehicle (the functional programming language Racket) to teach formal proof methods. The content spans topics such as equational reasoning, Boolean algebra, natural deduction, and induction used to prove correctness of programs. The Mathematical Foundations of Computer Science course was selected for its unique curriculum that blends programming with formal proof methods, making it an ideal setting for studying the intended skill transfer.

### 2.2 Isomorphic Questions

Zingaro and Porter [11] demonstrated the efficacy of the techniques of isomorphic questioning in a computer science context. We employ a similar method here. Students were given one week, during the first week of the course, to answer the following question:

*“Somewhere a bomb has been placed in a house on either Main, Oak, or Sunset Avenue. The street number of the house with the bomb is*

*either 100, 200, 300, 400, or 500. A bomb sniffing dog has already ruled out the following houses: 200 and 500 Main Ave, 100 300 and 500 Oak Ave, and 400 Sunset Ave. The FBI knows the avenue name of the bomb, and the SWAT team knows the street number (moreover, each agency is fully capable of logic and aware of all the information in this paragraph)*

*The FBI radios SWAT saying: “We don’t know the location of the bomb, and we are certain that you don’t either”*

*SWAT messages back: “We didn’t know until your message, but now we do”*

*To which the FBI replies: “ah, so we do too now – see you there!”*

*Both agencies arrived at the scene and were able to jointly defuse the bomb in time. Where was the bomb located? Explain your reasoning as best as you can.”*

Near the end of the course, over two months later, students were given a week to answer an isomorphic question (below). At no time in the intervening period were the problems ever discussed (other than to dismiss student requests for a solution), nor was material ever covered in the course that directly related to methods enabling the solving of these brainteasers. The feature matrix for this second problem is the same, but with two rows swapped, and two columns swamped.

*“George and Martha are married and perfect logicians and computer programmers but have both forgotten their marriage anniversary. However, they do both remember the following facts: [i] it happens on a month that begins with a “J”. [ii] The numerical date was a two-digit number divisible by 5. [iii] It does NOT coincide with any family member’s birthday [the birthdays are on January 15th, January 30th, June 20th, July 10th, July 25th, and July 30th]. George does correctly remember the month (but not the date), and Martha remembers the numerical date (but not the month).*

*George says to Martha: “I don’t know our anniversary, but I know you don’t either.” Martha replies: “I didn’t know until you said that, but now I do!” George then shouts: “Ah, because of that, I do too now!” Then the two go off to prepare their anniversary celebration, which happens to be the next day. What is the month and day of their anniversary? Explain your reasoning as best you can.”*

The author cannot express the solution better than this student’s submission (which also came accompanied with a nice grid drawing, not included here):

“It can’t be a 30, since then Martha would have known (June 30 is the only 30 option). Similarly, it can’t be in June, since then George couldn’t have confidently said Martha didn’t know (otherwise it might have been a 30). So Martha now knows June is out, but is still able to determine the anniversary, thus her number isn’t 20 (otherwise she wouldn’t have been able to determine Jan vs July). George knows this, so both 20 and June are ruled out, this leaves Jan 10, Jan 25, and July 15. Since George says he now knows, it couldn’t be a Jan month (or he couldn’t tell between 10 and 25), so the anniversary must be July 15.” The solution to the first problem is done analogously and is left as an exercise to the reader (hint: the answer is 200 Oak Ave)

These specific isomorphic questions were chosen for their ability to engage students in logical reasoning and problem-solving, closely mirroring the cognitive processes involved in proof writing.

### 2.3 Categorization

Precise grade distributions for students’ Calculus 1 and Intro to Programming were not available to us, but the authors did have access to all student transcripts so we did not have to rely on self-reporting. As a metric for performance in these courses, we divided students up as either earning an “A” or not. We kept the categorization to the minimum of just two groups, since fine tuning too much (e.g. A,B,C,D,F) would lower the quantities in each group and make the results too sensitive to random variation (not to mention the fact that often the distinction of an “average” student earning a B vs a C can be subjective based on the instructor, whereas there tends to be more absolute agreement on the extremes). Note that if a student tested out of a class (e.g. Advanced Placement in Calculus AB/BC or Computer Science A) or transferred in from another institution, or took the class Pass/Fail during COVID, then the transcript did not reflect a letter grade and this was registered in the “not an A” category. The exception to this is if we had no grade of record for Calculus 1, but saw that subsequently the student had earned an A in Calculus 2, then we did categorize them as “A” in Calculus (and similarly for the Intro to Programming course).

When it came to Math Foundations course that we were actually instructing, we did of course have full information of the entire gradebook, and so here we could be more judicious in categorizing the two groups. The students were divided into Lower Performers and High Performers, based on whether they scored below or above the median (83.2) for the class. For the coding of the question responses themselves, we also divided those into two groups: Undeveloped (generally weak responses), versus Developed (satisfactory answers, which indicated sound logical analysis, even if not a complete solution). The binary categorization into ‘Developed’ and ‘Undeveloped’ responses was

adopted to provide a clear and straightforward assessment of the students' logical reasoning abilities, which are central to both programming and proof writing. Although we kept to these two overall groups for purposes of statistical analysis, we also fine tuned each of those into three tiers in order to have a better measure of improvement between the two questions.

Specifically, the three tiers in the Undeveloped category (in increasing order) were: [i] no real attempt, [ii] providing some reasoning, but the rational was outside the bounds of the logic problem (see examples in next section), and [iii] identifying the value 500 [or in the isomorphic anniversary problem, the 30th] as being of special significance, but then just immediately (and erroneously) concluded that it must be the solution because of its uniqueness. The entry tier of the Developed category is when the student realizes that because of the uniqueness feature of the 500 that it actually cannot be a part of the solution, but then does not take the reasoning any further. To reach the next highest tier of the Developed category, the student then had to pivot to the other feature and rule it out as well. Very few students were able to proceed past that point in the logic, but all those that did were able to fully complete the solution which was the definition of the highest tier of the Developed category,

### 3 Results

#### 3.1 Sample Student Responses

A small number of students in the lowest tier categorization effectively left the problem unattempted and simply wrote “This problem cannot be solved”, or “the sentences between the two of them don't add any additional information”. There was a significant number that did express some reasoning, but that were outside the scope of the problem or made some illogical leaps. Some creative illustrations of these from the second lowest tier are: “terrorists would place a bomb at 300, in the center of the town, to maximize damage”; “they were speaking in code to each other. When they said ‘too’ that meant 200, and saying ‘see’ = sea = sunset”; “using probability formulas, there is a 1 in 15 chance that...”; “there is a pattern to how the dog moved. It would not have gone from 100 to 300 unless it didn't smell a bomb at 200”; “the FBI could have followed the SWAT team to find the street. Or bugged their headquarters to figure out the location” Similar “creative” responses for the Anniversary problem included: “the anniversary must be in January at the start of the year to give them enough time to forget”, or “a lot of weddings occur in June”. Samples from the highest tier still in the Undeveloped category are: “500 is the only street number of all the avenues in which two have been ruled out, thus the bomb must be there” An example of the entry tier in the Developed category would be: “The easiest location to rule out was 500, as if the bomb

was there then SWAT would have known immediately. . .”. If they had stopped there, that would kept them on the lowest rung of Developed, but in fact, the previous student entry then continued “. . . This also rules out sunset ave, since otherwise 500 would have been an option”. Since that ended their submission, that response was in the second highest category. An example categorized at the highest Developed level was already illustrated in section 2.2

### 3.2 Analysis of Findings

We succeeded in our goal of making the brainteaser challenging, as nearly three-quarters of the class gave Undeveloped responses in the first week of the course.

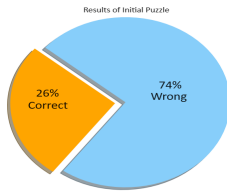


Figure 1: most students could not solve the initial puzzle

We wished to check if there was any correlation between their ability to do this sort of informal proof and their proficiency in programming. So, we looked at the submissions after grouping them based on their performance [A vs not A, which was about a 30/70 split] in their Intro to Programming class. A significant change in the Undeveloped/Developed distribution would be indicative of a link between programming and proof writing. And in fact, the results are:

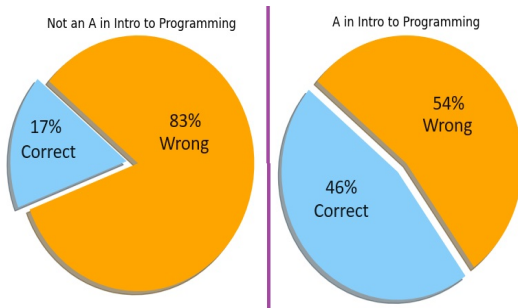


Figure 2: Puzzler Solvers partitioned by Programming grade

A  $\chi^2$  test of independence between performance in their Programming course and their performance on this proof writing task rejects the null (that they are independent) with a p-value  $\approx .0003$ . So this is a statistically significant difference between those two distributions. Initially, we were tempted to attribute this to merely “good scholarship”. That is to say, on this surface this could just be an appearance of a distinction, but really both are simply a measure of the academic strength of a student. In order to dispel that possibility, we also measured the other course that all our students had taken, Calculus 1, and used that as our control. If there really had not been something specific attributable to the programming course versus a math course, then we should see a similar distinction when we repeat the breakdown on these same students based on whether they made an A in Calculus 1 or not [about a 40/60 split]. Here is what we found:

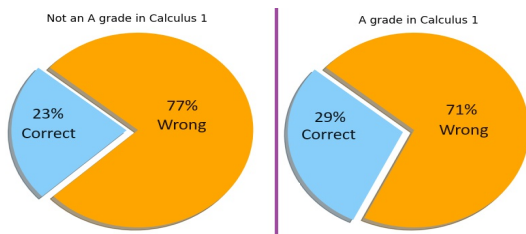


Figure 3: Puzzler Solvers partitioned by Calculus grade

There is no statistical difference between these two distributions at the  $\alpha = .05$  level. So in broad terms, calculus does not seem to have an impact on one’s ability to write proofs, but programming does!

Of course, there still remains the question of whether we can justify that this is actually a causal relationship. To address this, we turn to our isomorphic problem which was asked at the end of the course. It is true that the overall percentage of Developed answers increased, but that alone is not sufficient to claim causality. We went deeper into the more finely tuned coding of responses, and actually measured if there was an increase in their tier, and then we compared that to their performance in the Foundations course. If the Foundation course was not affecting their proof-writing, we would expect not to see a drastic difference between the before/after performances, or at least the rate of improvement would be the same regardless of their comprehension of the course material. Instead, see our actual findings. Note that since we are measuring improvement in a single-tailed test, students that had already scored in the highest tier the first time could not possibly have improved and



thus were exempt from this analysis. Additionally a few students withdrew from the course, hence the drop from  $n=43$  to  $n=35$ , (We did in fact have one anomalous student that inexplicably was able to solve the first puzzle but not the isomorphic one later). The results here show a marked distinction between the two pie charts ( $p\text{-value} \approx .003$ )

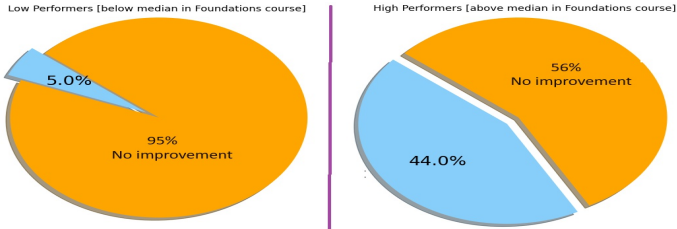


Figure 4: Puzzle improvement partitioned by grade in Foundations course

## 4 Conclusions

It has been known since 1969 that, at the purely theoretical level, there is a strong connection between proofs and programs: the Curry-Howard isomorphism [10] provides a mathematical demonstration that there is a one-to-one correspondence between models of computation and proof systems as formal objects. However, that is a far cry from establishing that proving and programming as human endeavors are linked from an educational perspective. This study makes some inroads to that case and clearly demonstrates linkage between those two skill sets. Moreover, this correlation is strongest when seen from the vantage point of a computer science class introducing logic, whereas the correlation to proof writing is virtually non-existent when compared to the standard Calculus class. The study indicates that students' computational thinking skills, which are critical to success in computer science, are aided by their basic programming experiences even more so than their standard mathematics sequence. This suggests that we should incorporate proof-building skills (such as case analysis, reasoning by contradiction, rejecting conjectures via counterexample construction) all throughout their computer science curriculum so that students can continually be learning it within a programming context, rather than relegating it to another department or isolating it all into one lone course. Perhaps even the common University prerequisite of Calculus for all computer programmers needs to be reconsidered.

## 4.1 Limitations

Even though we have over one hundred individuals involved in the study in the Fall, it is still only performed at our one university; it is conceivable that our institution might not be representative of undergraduate computer science majors overall. Also, these findings concentrate on three specific courses undertaken by students throughout the year; however, they were also enrolled in additional classes, possibly including other Math or CS courses, which might have played a role in enhancing their overall intellectual growth and development. Moreover, as we repeat the experiment, the likelihood of a security leak in which the solution gets out becomes greater. Furthermore, in the era of generative AI, the authenticity of students' submissions may be questioned, especially considering that they had a week outside of class to finish most of their assignments. Another issue is that while our results lean towards a causal link between the formal proof skills taught in our Foundations course and informal proof writing skills, what would be more relevant to most of us in the CS education community would be a causal link between learning proof and actual programming skills. And lastly, the connection between the capability to articulate a puzzle solution clearly and the transfer of these skills to formal proof writing and abstract concept reasoning remains unclear. Employing puzzle solving as a foundational step for proof writing is promising, yet it's uncertain whether this ensures a seamless transition of skills to more formalized proof writing scenarios.

## 4.2 Future Work

We plan to continue this study throughout the year with greater numbers; and to address some of the threats to validity mentioned above, we hope to involve other institutions as well. Moreover, there could be future experiments designed to show a direct causal link between learning how to write proofs and writing better programs. Specifically, we would like to drill down to examine individual logic operations and templates to see how they are impacted and get impacted by their programming counterparts. most non-trivial proofs in CS require induction, which is not mentioned in the paper. As the author(s) note, the theoretical correspondence between proving and programming extends to proofs by induction as well – those are just the logical equivalent of structurally-recursive algorithms. It would be very interesting to somehow see a connection between understanding recursion in a programming language and the ability to write proofs by induction. For instance, how do students view the implication operator in light of their experience with the If-Then procedural paradigm? How does their proficiency (or lack thereof) with recursively defined functions influence their facility in writing proofs by Induction? Could a better

understanding of one lead to a less steep learning curve for the other? Most non-trivial proofs in Computer Science necessitate the use of induction, a topic not addressed in this paper. The theoretical parallelism between proving and programming also encompasses proofs by induction, which are essentially the logical counterparts of structurally-recursive algorithms. We plan further studies to explore linkage between the comprehension of recursion in programming and the proficiency in crafting proofs by induction. Answering these questions could help inform our approach to teaching programming. We also hope to soon extend this study by implementing a software tool to aid students in constructing proofs and analyze what impacts this has on their programming abilities.

### **4.3 Acknowledgements**

The author expresses appreciation to all the students and instructors of the Mathematical Foundations of Computer Science course at Drexel University from 2020-24 for participating in this study.

## References

- [1] Joint Task Force on Computing Curricula. *Computer Science Curricula 2013*. New York, NY, USA: ACM Press., 2013. ISBN: 978-1-4503-2309-3. DOI: 10.1145/2534860. URL: <http://doi.acm.org/10.1145/2534860>.
- [2] Susanna S Epp. “The role of logic in teaching proof”. In: *The American Mathematical Monthly* 110.10 (2003), pp. 886–899. DOI: 10.2307/3647960. URL: <https://doi.org/10.2307/3647960>.
- [3] Mel Friske. “Teaching proofs: a lesson from software engineering”. In: *The American Mathematical Monthly* 92.2 (1985), pp. 142–144. DOI: 10.2307/2322651. URL: <https://doi.org/10.2307/2322651>.
- [4] Cruz Izu and Claudio Mirolo. “Learning Transfer in Novice Programmers: A Preliminary Study.” In: *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education 1* (2021), pp. 178–184. DOI: 10.1145/3430665.3456336. URL: <http://doi.acm.org/10.1145/3430665.3456336>.
- [5] Juan Pablo Mejia-Ramos and Matthew Inglis. “What are the argumentative activities associated with proof?” In: *Research in Mathematics Education* 11.1 (2009), pp. 77–78.
- [6] George Polya. *How to solve it: A new aspect of mathematical method*. 246. Princeton university press, 2004.
- [7] Edward A Scheinerman. *Mathematics: a discrete introduction*. Cengage Learning, 2012.
- [8] Ronny Scherer. “Learning from the past—the need for empirical evidence on the transfer effects of computer programming skills”. In: *Frontiers in psychology* 7 (2016), p. 1390.
- [9] Christian D Schunn and M Patchan. “An evaluation of accelerated learning in the CMU Open Learning Initiative course Logic & Proofs”. In: *Report, Learning Research and Development Center, University of Pittsburgh* (2009).
- [10] Richard Zach. “The significance of the curry-howard isomorphism”. In: *Philosophy of Logic and Mathematics*. De Gruyter, 2019, pp. 313–326. DOI: 10.1515/9783110657883-018. URL: <https://doi.org/10.1515/9783110657883-018>.
- [11] Daniel Zingaro and Leo Porter. “Tracking student learning from class to exam using isomorphic questions”. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 2015, pp. 356–361.

# Agile Practices and their Role in Enhancing Productivity in Student-Run Software Development Projects\*

Sandeep Mitra and Eric Rosenhein  
Department of Computing Sciences  
SUNY Brockport  
Brokport, NY 14420  
{smitra, erose3}@brockport.edu

## Abstract

This paper describes the manner in which the adoption of aspects of an Agile approach has been effective in managing student team projects of significant complexity in the context of a software engineering two-course sequence. Besides exposing students to a contemporary industrial practice, use of Agile techniques has enabled closer monitoring of student team progress, individual student contributions and facilitated better communication among team members. This paper discusses both the advantages of, and the challenges present in, adopting the Agile approach to its fullest extent in an undergraduate classroom. We present our observations of the effectiveness of adopting the Sprint/Scrum approach. Also, we discuss the challenges of practicing the concept of self-organizing teams, and adequately estimating required effort. We identify issues related to being Agile and delivering the desired product, such as the importance of discovering a potential effective Scrum master (who will function as the team leader) in pre-requisite courses. We then discuss supporting the development of soft skills related to effective teamwork and individual technical skill acquisition. We analyze reflections from students and team leaders gathered over the years and provide potential approaches to educating students better about Agile practices.

---

\*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

In the software engineering component of a typical Computer Science curriculum, a team project involving the development of a software product of significant complexity is usually required. Such projects are often considered to be “capstone” experiences, and, as we discuss in the “Related Work” section, numerous authors have discussed the way such projects should be managed by the instructor(s) involved. Over the last decade or so, since Agile methodologies became popular in industry, many authors discuss their experiences in bringing the teaching of Agile techniques into the classroom. Their goals mainly fall into two major categories – one, to see if such techniques enhance the chances of the student teams being successful in developing the software product, and two, to teach students a contemporary standard industrial practice, thus enabling them to compete better in their career search efforts. This paper discusses the use of Agile practices by student teams. At the authors’ home institution, students learn software engineering concepts and gain the needed hands-on project experiences in a two-course sequence. The essential goal of this course sequence is to provide students the opportunity to work on a “real world” project – i.e., a project requiring the development of a product that would satisfy a customer who would deploy it for day-to-day use at their business site. The project’s scope is dictated by what features the customer needs in the product to make it “minimally viable”. Therefore, meeting the viability requirement is necessary for students to feel that they have had a true “real world” experience that can substantially enhance their resumes. Failing to meet the minimal viability requirement will result in the effort not being characterized as a complete success, and negatively impact the final grade for the student/student teams.

Next, we note that the scope of these projects is such that it is essential for students to work in teams. The ability to be a good team player and demonstrate leadership/initiative in the context of teamwork is an important learning outcome of the course sequence and also of our ABET-accredited program. The organization and management of student teams, and the evaluation of individual contributions to the team, is a significant issue, and is the main thrust of this paper. Thirdly, it must be stated that we face certain constraints. Resource availability is limited, as we cannot have multiple full-time teachers dedicated to any of the classes, nor do we, currently, have qualified teaching assistants who can assist. Also, the curriculum is “tight” in the sense that it is challenging to add required credits to the curriculum (e.g., adding a 1-credit supervised lab component to any of the courses in the sequence would be difficult). Finally, we require that the code developed be reviewed for quality control – i.e., it must follow coding techniques that increase its extensibility and maintainability, to be able to handle perfective maintenance requests that

come in from the customers.

In the context of the above environment, this paper focuses on which aspects of Agile practices, if any, can provide a means for the instructor to manage the project teams. The primary objective is to achieve the goals (learning outcomes) of the two-course sequence, satisfy any external customer and provide a sense of accomplishment to students. The first author of this paper has been the instructor of record for this course sequence for over a decade and had taken a waterfall-like approach to project management until 2015, with a focus on rigorous modeling of the behavior and structure of the system to be developed, as described in [5]. A level of dissatisfaction, perceived in student reflections, with a “hands-off” approach to team management during the implementation phase led to the investigation and gradual incorporation of Agile practices for team management during this and later phases. This paper presents the results of this experience in numerous student-run projects over the last eight years.

## 2 Related Work

Over the last ten years or so, we have studied the experiences of various authors in setting up and managing student teams to work on projects that are part of software engineering-focused course(s). In this regard, two issues of importance are the facilitation, monitoring and evaluation of communication among the team members and the project customer (and/or instructor), as well as the ensuring of the accountability of individual team members in the team evaluation process. A detailed analysis about the building of effective multi-disciplinary teams is provided in [6], including an extensive review of related work on the subject. Important learnings from this experience are the need to get all stakeholders to agree on common, reasonable goals for the project, ensure that each team member has a vital role to play, make contributions from (groups of) team members visible, and educate students on the importance of valuing contributions from members with varied skills. This paper also mentions the value of good team organization and the need for quality leadership skills when seeking to develop a useful software product. In [11], the authors present a comprehensive discussion of assessment in “large-project courses”, with an in-depth presentation of various assessment techniques, and a focus on ensuring accountability via student peer evaluations, teacher observations, student reflective reports, targeted interviews of students, etc. The authors emphasize the need for formative assessment (i.e., early feedback that seeks to improve later performance) and summative assessment at the end of the project that should incorporate both group and individual components.

Upon receiving advice from our own stakeholders (including alumni) about better team management, we began exploring fellow instructors’ experiences

with Agile practices ([1, 9, 12, 4, 2, 3, 8, 10, 7]). Not all instructors discuss the use of Agile for software development purposes. [8], for example, focuses mainly on using Agile as a vehicle for teaching project management techniques, and [7] takes a “low code” approach to accommodate a short six-week course offering. All authors take a Sprint/Scrum approach. The length of the course’s term(s) affects the number of sprints that can be scheduled. The limited number of class meetings per week require the daily Scrum meetings in industry to be carried out as 1-2 in-person meetings during class (in which the instructor(s) may act as project mentor(s)), and remote meetings, which may be mandated to be conducted on a certain tool so the instructor may monitor them ([1]). All authors use the standard Agile artifacts of the Product Backlog (where the features yet to be implemented till project completion are identified) and the Sprint Backlog (where the implementation, testing and integration tasks to be tackled during the upcoming sprint are outlined). The role of Product Owner manages the Product Backlog and may be played by the instructor(s) ([2, 8]) or students ([1, 3]). The role of Scrum Master who manages the Sprint Backlogs may be played by the instructor(s) ([4, 2]), students ([1, 8, 12, 7]) or shared between the two ([10]). The need to provide leadership opportunities to more students leads to some instructors rotating the abovementioned roles ([1, 8, 10]).

While discussing the tasks (user stories and “story points” in Agile terminology) that need to be included in a particular Sprint Backlog, instructors attempt to empower the (inexperienced) student teams to estimate efforts by themselves, by providing guidelines such as “one story point unit of complexity equals one hour of lab time” ([12]). Allocating these tasks to individuals within the team leads some instructors to discuss their emphasis on incorporating the Agile practice of “self-organizing teams” into their approaches ([1, 12, 2]), with varying degrees of success. The actual conduct of the activities associated with each Sprint leads some instructors to explicitly mention that students volunteer for these tasks which are then tracked on a task board ([12, 10]). It is worth noting that many instructors mention that the more experienced members (i.e., instructors/ TAs) act as consultants to provide help on the Agile practices themselves as well as assist with questions on the tools and technologies used ([1, 3]). Each Sprint ends with a demo (explicitly mentioned in [1, 9, 10, 7]), a Sprint review conducted mainly by the student team themselves, and a more detailed Sprint retrospective is often conducted, with instructor/ TA help, to provide feedback on what could be done better in future Sprints. Following these, the Backlog for the next Sprint is created appropriately. In this context, the important role played by appropriate tools for enhancing team productivity is mentioned by many – e.g., tracking tools (The use of Trello is mentioned in [3] and [10]), version control tools like Git/GitHub, and other



project environment-specific tools like Mendix in the “low code” approach ( [7]) are discussed.

It is important to note that instructors discuss the pre-requisite structure leading up to the main project class(es), as well as the instructional (lecture) content of the early weeks of these classes to ensure that students have adequate training for conducting the project. Finally, while discussing the results of taking this approach, instructors mainly discuss student satisfaction with the experience of undergoing the experience of a contemporary industrial practice, especially in the context of project management and teamwork ( [9, 12, 8, 7]). Some discuss the extent of success in the development of necessary features in the final software product ( [4, 3]).

### 3 Characteristics of our Pedagogical Approach

Our two-course software engineering sequence takes a model-centric approach to software development as described in [5] . Students create UML (Unified Modeling Language) behavior models (e.g., sequence diagrams) and structural models (e.g., class diagrams/ CRC (Class-Responsibility-Collaboration) cards) in the first course geared to developing a system conforming to the classic three-tier (MVC: Model-View-Controller) architecture. As discussed in [5], this approach enables students to see the “big picture” of the system under development, and thus lays the basis for students to write extensible and maintainable code. The actual realization of the modeled system is done in the second course. In this course, relevant technologies/ tools are taught during the first six weeks of the semester, and students work in pairs on small exercises to familiarize themselves with these. The project is then worked on for the last 7-8 weeks of the semester. Prior to 2015, we used a waterfall-like approach. Students organized themselves into teams ranging in size from four to six. They were required to present their progress in completing the implementation, testing and integration work at three points during this 7–8-week period. The instructor acted as Product Owner, and indicated the features that each team would have to implement and present (i.e., demonstrate the working of) at each of these three points. Each team had the option to adjust what they must present depending on what they had completed to an acceptable level of satisfaction at the previous presentation point, and, therefore, needed to improve for this demonstration. After each presentation, each team was provided with feedback about the quality of their work and given suggestions for improvement. It is important to note that the instructor reviewed code, provided suggestions on refactoring it to improve quality and compliance with standards and was available for help with the technology/tools, but did not manage the team in any way. The students were responsible for deciding upon tasks, allocating

these tasks among themselves and monitoring progress. A large percentage of the final grade was based on the quality of the final product, and, after delivery, students completed peer reviews of their team-mates. Using an average score from these reviews, the instructor assigned the rest of the grade. Reflections from students indicated considerable dissatisfaction with this process, even from successful teams. We noted that many students indicated that it was “heroic effort” on the part of some team members that led to success, and the grading scheme did not penalize non-performers to the extent deserved, due to two main factors – one, being a good “team citizen” did not carry enough weight in the final grade, and two, “student solidarity” evident in the peer reviews often resulted in these non-performers not being graded appropriately, especially after successful delivery.

To rectify the situation, practices from the Agile approach were incorporated after 2015. In this paper, we discuss our experiences in adopting this approach in the following semesters (student enrollment numbers in parentheses): Spring 2016 (34 students), Spring 2017 (36), Spring 2018 (25), Spring 2019 (23), Spring 2021 (18), Spring 2022 (13), and Spring 2023 (24) – we are excluding the “COVID-chaos” affected Spring 2020 semester. In addition, the first author was the mentor of two “student club” teams in Spring 2017 and Spring 2019 that sought to develop products for two real-world customers. In these two projects, the five-member student teams took an Agile approach (i.e., used Sprint/ Scrum, self-organized themselves and estimated required effort by consensus). It must be noted that they carried out the necessary design and technology/tool learning tasks by themselves, without mentor intervention. Neither of these projects could meet the minimum viability requirement for deployment by the customer. Interviews with the teams indicated that the reasons for failure included the time taken to learn the unfamiliar but needed technologies/ tools, leading to “bad” estimates of effort, and the need to revise the designs “on the fly” to meet changing customer requirements. Therefore, despite the Agile philosophy focusing on working code and “discouraging” a model-centric approach, we, in our course sequence, still focus on modeling and supervised learning of needed technologies, leaving the same 7–8-week period we had available when we took a more waterfall approach, for actual development work.

Since Spring 2016, students have been required to get their team members approved by the instructor, who sometimes re-balanced the team personnel to have a mix of abilities in each team. This was usually determined by grades and observations in earlier “extensive programming” courses. During the development period, the instructor acted as Product Owner and announced the set of features that had to be presented at each of the three “epics” whose dates were announced at the start of the project. Student teams were asked to elect

a Scrum Master, a role that could be rotated among the members after each epic, at the team's discretion. Each epic was expected to be composed of a number of Sprints. The Sprints themselves were 7-9 calendar days long. Since the class met for 1.25-hour sessions twice each week, there were at least two Scrum meetings that were to be conducted in-person at the start of each class session. Teams were strongly encouraged to have more frequent Scrum sessions outside of class hours, possibly remotely using a communication tool of their choice. The instructor offered to provide input on creating each Sprint Backlog, but the students collectively decided on its content and task allocation among the members. At the end of each Sprint, teams were asked to conduct a review to determine whether they were making good progress and were encouraged to have a representative consult with the instructor about any lack of expected progress, analyze causes (especially those arising due to non-performing team members) and receive recommendations for the subsequent Sprint. For each epic, 70% of the grade for each student was based on the extent to which the team met the epic's feature implementation requirements, as well as code quality (determined by instructor review of appropriate code samples). 30% was based on peer-evaluation, where the extent of each student's contribution, and more importantly, effective communication was to be used as a basis for judgement. There were no mandates to use any tools for development, testing and communication, except for Git/GitHub (version control). Students were required to provide reflective feedback on the process, as discussed in the next section. During the 2016-2019 period, the use of this approach enabled at least one team each semester to deliver a minimally viable product. Out of 22 teams observed for this period, there were only two that failed to deliver even 25% of the full set of features needed to get a passing grade.

Based largely on student feedback, the above approach was modified during the 2021-2023 period to lay a greater emphasis on team leadership. The instructor identified a set of students who would act as Scrum Masters, who would serve as team leaders. This selection was based on their academic track record, including the instructor's observations of their ability in prior classes. This was not a role to be rotated, except under extraordinary circumstances. The leader was required to interface with the instructor about the content of each Sprint Backlog. Also, the leader was required to conduct a Sprint retrospective with the instructor. Task allocation was left to each leader. During each retrospective, the instructor required information about non-performers, and discussed attempts made to enhance communication with these individuals. Also, the instructor sought input on technical challenges perceived as being faced by the non-performers and discussed possible means of providing extra help. The grading scheme was modified to reduce the component based on meeting deliverables and code quality to 55%. For the remainder of the grade,

the instructor kept notes from the leader’s input during Sprint retrospectives, which led to recommendations for improvement made to the student (either by the leader and/or the instructor) to the student. The extent to which the student complied with these recommendations and improved performance during subsequent Sprints determined the actual grade. Of the 12 teams observed during this period, no team failed to get a passing grade.

## 4 Student Reflections and Instructor Observations: First Phase

During the 2016-2019 period, students were asked to reflect on the effectiveness of the Agile-oriented approach to team management by considering the following four issues:

- (i) Did the more “fine-grained” phases (i.e., the 7–9-day Sprints) help the team to stay on track better, and help accomplish what was expected at each required presentation?
- (ii) Did the fact that the teams themselves allocated tasks to members each Sprint, and frequently monitored progress (Scrums) – with the instructor available as a consultant if needed – help to enhance team “velocity” ([1]) (aka productivity)?
- (iii) Did the preparatory exercises (modeling the system, learning the features of the needed implementation frameworks) prepare you to be productive in the implementation work, given the schedule?
- (iv) Was the grading scheme fair, and do you have any overall suggestions for improvement?

Students were given a small incentive – a 1% credit in the final grade – to submit their reflections. 97 out of 118 students chose to provide input. Almost all students (95/97) appreciated the preparation in the form of a semester-long modeling project and the teaching of the features of the technologies to be used, even though it limited the time for actual development to 7-8 weeks. The way we ensure that the model-centric approach is useful for development is described in detail in [5], so this result was not surprising. Also, all responding students (97/97) appreciated the “fine-grained” Sprint/Scrum approach. However, it was evident that the effectiveness was heavily influenced by the nature of the student leadership. A student Scrum Master who kept in near-constant communication with the team, even on days that class did not meet, using remote communication tools (e-mail/Discord/etc.) was able to make the team more productive. At the very least, such a leader could correctly identify the

non-performers (“social loafers” as described in [8]) early and provide them with opportunities to deliver on their assigned tasks. Such opportunities could include the provision of extra help from other team members and/or the instructor. On the other hand, a more “easy going” Scrum Master was seen as one who did not communicate very much. Some students saw such a Scrum Master as one who lacked motivation themselves, were possibly content with a lower grade, and this was detrimental for team velocity. Consequently, the more motivated student(s) would usually put in “heroic efforts” on their own to deliver on the tasks in the Sprint Backlogs. This resulted in the desire to replace the individual in the Scrum Master/ team leader role. It was this experience that resulted in the instructor, over this period, receiving requests to intervene and change leadership (“I am actually the Scrum Master even though I was not elected to this role and did not really want to do it” was a frequent refrain heard by the instructor). Also, due to this experience, some students (15/97) explicitly mentioned that they were opposed to the rotation of the Scrum Master role (“If we have a good Scrum Master, keep them for the whole project”). Also, 33/97 students stated that they wanted the instructor to take on the role of Scrum Master in addition to Product Owner, because of their perception that as they did not have enough prior experience working with those in their team, they were not best suited to taking on a leadership role – i.e., figuring out and assigning tasks to their team-mates. These students also said they felt that they did not have enough “authority” to make non-performers improve, as the grading scheme involved a “full-fledged” peer evaluation, with the leader’s input given the same weight as every other member of the team, including other “non-performers”. In our resource-constrained environment, this was not feasible, unless class enrollment was significantly restricted. Therefore, in the period 2021-2023, a new approach was taken (see section above), with the instructor choosing a set of team leaders.

## 5 Student Reflections and Instructor Observations: Second Phase

With a greater emphasis on team leadership during the 2021-2023 period, this section addresses the way reflections were sought from team leaders and how these were analyzed. Leaders were asked to reflect on a set of questions in the following categories:

- (i) Accountability of team members
- (ii) Ensuring minimal viability of the software product under development
- (iii) Use of modeling for trainee software engineers and

- (iv) a “General” category in which they were asked to focus on the means of communication used by the teams, and how to enhance the effectiveness of such communication, plus the adequacy of resources provided (in terms of time and available experienced personnel).

Detailed reflections were obtained from seven team leaders, and the authors were able to schedule in-person follow-up interviews with four of them. In the first category – i.e., enhancing accountability – the interviewees resoundingly supported the idea that grades are ultimately the only real leverage in keeping team members accountable for their work. In this regard, there was mention that unmotivated students will exist, despite the efforts of those working with them. Few solutions to this problem were offered, but one suggestion stood out. Instead of the current model where student team leaders suggest grades of their group members, there must be a way in which individual assignments from the Sprint backlogs are assigned and graded by the instructor, stemming from meetings with the team leaders. While the details were not discussed in depth during these interviews, the goal of this solution is to have the “look and feel of a group project, but the accountability of an individual assignment.”

Various approaches were used by team leaders to manage deadlines and splitting up the workload. Teams setting hard deadlines utilized strict enforcement of standards for working, mergeable code often based off a “code skeleton” created by the team leader. This method proved effective, but team members still did not always contribute enough to meet these expectations. To remedy this, these teams deployed various strategies of using the grade as a means of enforcing accountability – an approach that was broadly successful. Essentially, these boiled down to assigning a final grade on the date of any set deadlines with little flexibility. Not too surprisingly, this negatively impacted team members participation in some cases. Some teams were observed to have an attitude of “get what you can done”. This method of using soft deadlines resulted in a lack of organization and only a select few from the team delivered on these deadlines. Seemingly the most effective approach was using a series of soft deadlines, set before the hard deadlines (usually, the epics), to act as buffers for unexpected scenarios. These also improved communication within teams that employed this strategy. Constant feedback led to more organized team leaders and more opportunity to reallocate resources in a timely fashion.

No matter the approach, across the board there needs to be clear expectations of group members and frequent communication by both team leaders and team members to maintain steady progress. Without these two group characteristics, the priority of the project in group members’ minds, and, consequently, available work time will slip away requiring last ditch efforts to ensure hard deadlines set in the syllabus are met.

In the second category – i.e., ensuring minimal viability of the software

product under development – most notably, six of seven interviewees described substantial extra efforts made by them to ensure they met the requirements at each deadline. This frequently involved debugging (many taking this course are not proficient enough), picking up extra slack where group members were not accountable, and/or often refactoring code. It has become evident through these discussions that many team members lacked the commitment and effort required to complete any extra features beyond the original project scope to a satisfactory degree, if at all. Using grades as a tool, awarding extra credit to individuals/teams that go above and beyond the requirements was a recommendation from all team leaders. One suggested that this policy, combined with many short deadlines, as described in the flexible approach above, could help build dedication to the final product within the team. Other methods as described in the first category (proper approach to assignments based on team composition, timely and clear communication, proper allocation of resources, etc.) can be boiled down to the need for proper team leaders. One must be diligent and committed to the task at hand to ensure their team meets their goals. It was common for team leaders to approach the instructor in a timely manner with questions to overcome “blockages”. On the other hand, team members were rarely observed seeking help from the instructor outside of class, despite a wide range of availability that was advertised during every class. Finally, it was mentioned that the sample assignments given during pair programming assignments (see Section 3, paragraph 1) were especially helpful in the eyes of all seven team leaders. It provided them and their teammates with the skills and samples (from the tools/technologies needed for implementation) to, at the very least, create a minimally viable product during the allotted time.

In the third category – i.e., use of modeling and preparatory exercises for trainee software engineers – all shared the thought that the pair programming assignments and the modeling and architecture (from the first course) prepared them better for this term project. Leaders were asked: If you were given the full semester to complete this project (without the prior exercises), would your group have been able to reach their goals? One of them responded, “most assuredly not”. This sentiment was felt across the board.

In the fourth category – i.e., general – most felt there was enough class time to complete the project, but an extra hour for lab would be desirable. As it stands, team leaders described the need for an additional two to six hours of additional work per week, largely depending on the team. Teams met anywhere from one or two times outside of class, coordinating through use of Discord or text. On top of additional lab time, all seven participants said a teaching assistant (TA) would surely help, but opinions on the matter still differed. One felt the TA would be helping mainly with coding basics the curriculum failed to ingrain, in all the student members, despite this being an

upper-level course. Another presumed it may not be a great advantage, given the instructor was available most of the time to begin with. The remaining five shared a similar idea, commenting along the following lines: “It would be helpful, but that can’t replace the need for proper preparatory work”.

## 6 Analysis of Reflections

An analysis of the abovementioned reflections indicates that enforcing accountability of individual team members is a significant issue. The Sprint/Scrum approach, which is a finer-grained approach than that taken in the pre-Agile (waterfall) days, facilitates the monitoring of team (and team member) progress more closely. From the above reflections, it appears that the instructor-in-charge should be more involved in this monitoring, and in the grading process. But this is challenging in a resource-constrained (single instructor only) environment. One approach to consider would be to empower the team leaders to (a) use detailed instructor-provided criteria to recommend a grade for each team member, (b) provide corrective feedback to the “non-motivated” team member(s) to seek additional help, from within the team and outside, and (c) require the monitoring of the team member about the extent to which they sought such help. To facilitate this process, the instructor must require the team leader to keep detailed records of the communication between the member and the leader. The instructor must also set up a course-specific grade appeal process that can be exercised at each epic.

Through the instructor’s own observations of teamwork, we believe that accountability requirements may be easier for a team member to meet if they are assigned a task which they are “comfortable” with. Agile sets the goal of having self-organizing teams, in which students decide among themselves the tasks they will take on during the Sprint (see [12, 2, 8] for their experiences). Our experiences indicate that in an environment where most of the effort in each Sprint/epic phase must be the writing of quality code, each student in the team must take on some coding task(s). Some students may prefer writing the front-end code, others may prefer to focus more on the middle-tier (business logic) and the back end (database) interface. We have noted that it is best if the team leader observes this in the earlier Sprints, and then recommends assignment of these tasks to the appropriate members in later Sprints – of course, the ability of the leader to accurately gauge this is an issue. Inexperienced students typically don’t “know themselves” well enough to know “what they are good at”. Later in the project, realizing what they are good at is a source of satisfaction for them. This, however, is sometimes coupled with the desire of the student to “branch out” and practice what may not be their forte now, and this conflicts with the need to utilize the available personnel best to



meet deadlines.

In order to ensure that a viable product is delivered at the end the 7–8-week project period, we note that the above reflections indicate that creating and documenting a design model of the system to be developed in the first course is considered useful. Also, practicing the use of required tools and APIs in the initial weeks of the second course (in pairs) is considered a necessary familiarization exercise. Although requiring that the student teams learn these on their own as part of project work would, in a sense, give “more” time to work on implementing project features, the majority of team leaders felt that this approach would decrease productivity. As it is, the student teams needed to learn Git/GitHub on their own, and several needed some time to practice use of this tool. In fact, several students informally requested that these tools be taught before the project launch, and practice exercises provided by the instructor. The lack of time in the semester to accommodate this request has prevented it from being complied with. Also, it is worth repeating that, as mentioned in Section 3 above, the instructor supervised two “student club” projects in 2017-19 where the students had to learn technologies the instructor was not familiar with, and then use them to develop a product on a client-sponsored project. Both these projects were not deployed, thus effectively failing. This is another data point to indicate that students must be trained to have the necessary background/experience (“comfort level”) with the needed technologies/tools before launching the actual project. We note that some authors ([9, 12, 7]) who require the students to learn the needed technologies/tools as part of the Agile approach (i.e., during project work) report “mixed” results. For example, one author states that students chose to use simpler (i.e., more familiar) technologies. Others design projects that use technologies taught in pre-requisites ([1, 4, 3]) have resulted in greater success with student deliverables. We note that [3] describes their project as part of a four-semester software engineering curriculum. [4] reports, as part of their “lessons learned” experience, that their students sometimes missed a more detailed “up-front” design, because using this they would feel safer about having taken the right direction towards the final outcome.

In order to ensure proper communication among team members, a typical student who takes this two-course sequence must develop a culture to work outside of class. It is essential that students also develop an attitude of responding to other students in a timely fashion. They must attend each class session and show up for remote Scrum meetings (being sure to inform the team if they have legitimate reasons for not being there). Team leaders must note the extent of their engagement in each meeting.

In the context of adequacy of resources, availability of “extra help” to make up for gaps in a student’s knowledge would be useful. It is clear that at

least some students felt somewhat unprepared for the challenges thrown up by project work – examples include training in Git/GitHub and more experience in debugging a (reasonably) large body of code. Additional instructional resources (tutors/TAs) would be helpful. More importantly – as some team leaders emphasized – students must be engaged enough to seek out and utilize any available resources.

## 7 Conclusions and Future Work

The Agile approach provides a framework for better monitoring of teamwork to the instructor, which the original waterfall approach did not. Use of Sprint-/Scrum enables shorter phases, with major demos occurring at the end of epics. In collaboration with the team leaders, the instructor can monitor individual team member performance more closely, and obtain a basis for encouragement of engaged students, while also having a basis for providing suggestions for improvement to under-performing students. The frequent engagement of the instructor with code review also provides them an opportunity to gauge the effectiveness of their teaching of necessary pre-requisites. At the same time, it cannot be denied that the use of this approach results in the need for greater commitment from the instructor, thus impacting their workload.

The experience of using the Agile approach in a curriculum where the teaching of Software Engineering is confined to a 2-course sequence indicates that enabling the students to fully learn and see the benefits of an Agile approach can be challenging. We have seen that the Sprint/Scrum approach can be adopted by student teams fairly easily. It is more challenging to enable student teams to effectively perform two important aspects of Agile – i.e., devising accurate estimates of the effort required to complete tasks in the backlogs, and constituting student members into self-organizing teams – in the time available. Enabling successful projects, therefore, requires finding good team leaders, and training them appropriately in developing leadership skills.

As part of our future work, we will seek to examine approaches using which we can identify quality team leaders early in the curriculum. Once identified, we need to train them in leadership skills. As a first step, the first author, who regularly teaches classes in the introductory programming sequence seeks to identify engaged (and high performing) students as potential leaders. The second author is in the process of developing a “leadership handbook” based on their experience as a team leader in these projects, and also basing it on their own leadership training received as part of extra-curricular activities. Means of “empowering” team leaders need to be practiced and observed for efficacy. For example, ensuring team leaders know the resources available to execute the project successfully and guide team members to access and utilize these

resources effectively is necessary. The goal is to create an environment where students taking these courses understand, from the early part of the sequence, that individual effort is necessary. At the same time, students must be of a team-oriented mindset where they realize the need to “learn much on their own”, so they can be a net contributor to the team’s progress.

Effective as our latest approach has been towards delivering the minimal viable product on time, we perceive a drawback in the sense that success is predicated on having an effective team leader through the course of the project. In order to be able to rotate the leader role, it appears that the cohort of students who have been through this experience need further opportunities within the curriculum to work on another team project of similar, or even increased, complexity. It would be beneficial if the curriculum could be expanded to include – say – a year-long capstone project. Within the context of such an opportunity, students could experience other Agile aspects such as self-organizing teams (in which the leadership role rotates per Sprint/epic), estimating effort needed and learning necessary technologies on their own while making progress. While our curriculum currently does not provide the opportunity for such a capstone project, we have supervised independent study projects in which a reasonably sized team (typically, 4-5 students who have been through this course sequence before their senior year) have adopted the Agile approach and delivered software products. In the course of these projects, the instructor of record has only acted as a consultant, and not taken any supervisory role. It would be our endeavor to see how this full capstone opportunity can be expanded to more students at our institution, and thus, hopefully, provide them more solid experience with a contemporary industrial practice.

## References

- [1] Vladan Devedžić et al. “Teaching agile software development: A case study”. In: *IEEE transactions on Education* 54.2 (2010), pp. 273–278.
- [2] Stoney Jackson and Heidi Ellis. “Supporting HFOSS using scrum in a capstone course”. In: *ACM SIGCAS Computers and Society* 45.2 (2015), pp. 36–37.
- [3] Martin Kropp, Andreas Meier, and Robert Biddle. “Teaching agile collaboration skills in the classroom”. In: *2016 IEEE 29th international conference on software engineering education and training (CSEET)*. IEEE, 2016, pp. 118–127.
- [4] Viljan Mahnic. “A capstone course on agile software development using scrum”. In: *IEEE Transactions on education* 55.1 (2011), pp. 99–106.

- [5] Sandeep Mitra. “Using UML modeling to facilitate three-tier architecture projects in software engineering courses”. In: *ACM Transactions on Computing Education (TOCE)* 14.3 (2014), pp. 1–31.
- [6] Robert Pastel et al. “Team building in multidisciplinary client-sponsored project courses”. In: *ACM Transactions on Computing Education (TOCE)* 15.4 (2015), pp. 1–23.
- [7] Laura Poe and Lionel Mew. “The effects of using the agile methodology as an instructional format for software development courses”. In: *Industry and Higher Education* 36.5 (2022), pp. 638–646.
- [8] Daniel E Rush and Amy J Connolly. “An agile framework for teaching with scrum in the IT project management classroom”. In: *Journal of Information Systems Education* (2020).
- [9] Tucker Smith, Kendra ML Cooper, and C Shaun Longstreet. “Software engineering senior design course: experiences with agile game development in a capstone project”. In: *Proceedings of the 1st International Workshop on Games and Software Engineering*. 2011, pp. 9–12.
- [10] Ben Tribelhorn and Andrew M Nuxoll. “Using agile and active learning in software development curriculum”. In: *2021 ASEE Virtual Annual Conference Content Access*. 2021.
- [11] Maria Vasilevskaya, David Broman, and Kristian Sandahl. “Assessing large-project courses: model, activities, and lessons learned”. In: *ACM Transactions on Computing Education (TOCE)* 15.4 (2015), pp. 1–30.
- [12] Ramrao Wagh. “Using scrum for software engineering class projects”. In: *2012 Agile India*. IEEE. 2012, pp. 68–71.

# GraySim: An OS Scheduling Simulator\*

Sierra Zoe Bennett-Manke and Maria R. Ebling  
Department of Electrical Engineering & Computer Science  
United States Military Academy  
West Point, NY 10996  
`{sierrazoe.bennettmanke,maria.ebling}@westpoint.edu`

## Abstract

A traditional course in Operating Systems includes the topic of CPU scheduling with a discussion of several different scheduling policies. Historically, we have used a paper-based worksheet to help students understand how different scheduling policies work; in contrast, this article describes a simulation that allows students to practice and receive dynamic feedback on their understanding of five common scheduling policies. The simulation, called GraySim, allows students to specify which one of the policies they would like to practice and the process configurations they wish to run. After inputting their answers into the user interface, the student can view the correct solution, as well as view feedback on their solution if it was incorrect. We also present the results of a qualitative study evaluating this simulation and its potential for helping students gain a better understanding of different scheduling policies. Our study involved showing students from the current cohort of Operating Systems students a demonstration of the simulator, allowing them to change the policy and process configurations, and then asking them to fill out a brief survey to document their perceptions of GraySim's utility, including its strengths and areas in which it needs improvement. We close the paper with a summary of these results and a discussion of future research.

---

\*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

Operating Systems is a challenging class, both to teach and to learn because complex, abstract topics are discussed on a regular basis. Students have little prior knowledge of these topics and frequently struggle to learn the details of specific policies. In addition, it can be difficult for instructors to demonstrate several of these topics due to the gap between abstract concepts and concrete implementations accessible to students.

To combat these challenges, some instructors have advocated using simulations to guide student learning through these topics, both through demonstrations in the classroom and exercises or labs that can be assigned as homework. One such professor, Steven Robbins, published a series of papers describing several different simulations and promoting them to the general Operating Systems education community. In his most recent paper, published in 2008, he specifically mentioned using these simulations and their contribution to student learning and performance. However, present-day instructors will find it difficult to leverage these simulations in their own classrooms because only one of the simulations is readily available on the Internet today.

In this paper, we describe a CPU scheduling simulation, called GraySim, that allows students to practice their understanding of how the CPU will schedule different processes. Once a student completes an exercise, they can check their answer and GraySim gives them feedback about any errors it has identified. We anticipate using this simulation during the Fall 2024 offering of Operating Systems at our institution. We plan to compare the performance of students during the upcoming course offering to that of students enrolled in the Fall 2023, which was before students had access to GraySim. In preparation for deploying this tool, we wanted to gather input from the current Operating System students to get their perceptions and suggestions about GraySim. This paper reports on the results of this qualitative study.

The rest of the paper is organized as follows. Section 2 discusses related work, differentiating our own simulation from those discussed in the literature. Section 3 provides the Operating System background needed to understand the simulation. Section 4 is a description of GraySim and how it operates. Section 5 is an evaluation of the qualitative results collected on GraySim. Section 6 discusses these results and our plans for future work. Finally, section 7 concludes the paper.

## 2 Related Works

Current OS students report using a simulation, called CPU Scheduling Visualizer, available in GitHub [1]. This simulation allows students to configure

jobs with arrival times, burst (aka service) times, and priorities. Students then select which scheduling policy to observe. They can watch the ready queue change in real time and can also see a chart that shows the order of jobs running on the CPU. Unfortunately, students are not able to test their understanding with the simulator and do not receive feedback about their mistakes, one of the main goals of our work.

Examining the literature, we find a series of articles by Steven Robbins, a retired professor of Computer Science at the University of Texas at San Antonio, between the years of 1998 and 2008 that progressively discuss a series of simulations to aid in the teaching of undergraduate operating systems. The different subjects of simulators he writes about are: process scheduling[16], synchronization[15], process interaction in Unix[17], disk head scheduling[11], address translation[14], concurrent I/O in Unix[13]. The final paper in this series[12] discusses his approach to teaching operating systems which included using all of these simulators. In this paper, he reports that students enjoyed using his various simulations and that one of the simulations improved student performance on related exam questions, but he does not provide the results of a formal study.

Of these simulations, the most relevant to our work is the 1998 paper that focuses on scheduling policies[16]. Robbins' scheduling simulation allows students to compare the performance of different scheduling policies more readily than our simulation does. In contrast, our simulation helps students better understand the policies and how they work in practice.

Peter DeRosa and his colleagues describe the design and implementation of the Vesper disk drive simulator[3] which provides a high degree of performance realism, allowing students to explore the performance consequences of their designs. DeRosa and his colleagues' goal is to provide sufficiently realistic performance so that students can understand the consequences of various design decisions related to disk I/O schedulers. Our work focuses instead on the mechanics of learning the policies, not on the performance realism.

John Dickinson describes using a simple hardware simulator to teach undergraduate students about operating systems[5]. Using this hardware simulator, he has his students build a simple operating system. Our work differs from that of Dickinson in that our simulator is focused on helping students understand a single topic, process scheduling, rather than providing the basis for building a simple operating system in its entirety.

Luiz Maia and his colleagues discuss using the SOsim graphical simulator[9] to teach operating systems. The basic idea is to provide a simple, visual tool that allows students to see an animated view of what is happening on the computer. Their paper also presents the results of a user study which asked for feedback from a cohort of students. The SOsim work provides a general

purpose graphical tool that can be used for a broad range of topics. In contrast, our focus is on a simple tool that helps students learn one specific topic in depth and provides students with concrete feedback.

Leo Paschoal and his colleagues present a simulator, called SSP-Edu[10], that allows students to select the type of computational system, define the scheduling policy, manage processes, run SSP-Edu with the given inputs, and then export the simulation data as a PDF after running it. SSP-Edu is open-source and accessible via GitHub. Although their students show improved performance after using the simulator, the sample size is too small to provide conclusive evidence. A major difference between GraySim and SSP-Edu is that GraySim allows students to select their solution and check it against the software-produced answer, whereas the SSP-Edu simulator simply provides the answer. SSP-Edu, however, also shows students the waiting time, turnaround time, as well as the averages for those times and throughput, which our simulator does not do. This feature helps students evaluate the performance of different scheduling policies.

Neeto Jain and P.V. Suresh develop a simulation to visually explain various scheduling policies[8]. Similar to SSP-Edu, students do not get to practice their understanding of the policies, they are given the answer and must analyze it themselves. In addition, their simulation starts every process at time  $t=0$ . This limits students' ability to understand how process arrivals interact with scheduling policies. These researchers did not conduct any evaluation of how much assistance the simulation provided to students, though their students reported finding the simulation interesting to use.

These simulations and approaches to teaching aim to close the gap between abstract concepts and actual practice. The goal of our work, much like that of the work described above, is to provide hands-on experience with operating system concepts to aid in student understanding.

### 3 Background

Uniprocessor scheduling is a common topic taught in Operating Systems. Five common uniprocessor scheduling policies[2] taught as part of this topic are:

- **First In, First Out (aka First Come, First Served)** is a non-preemptive policy that selects the next job to run based on which job has been awaiting service the longest.
- **Shortest Job First** is a non-preemptive policy that selects the next job based on which job has the shortest expected service time.
- **Shortest Time-to-Completion First** is a policy that selects the next



job based on which job has the shortest remaining time to completion; it is preemptive at the granularity of job arrival.

- **Round Robin (aka Time-slicing)** is a preemptive policy that cycles through jobs, giving each a quantum of compute time; the quantum can vary.
- **Multi-level Feedback Queue** is a preemptive policy that has multiple queues, each assigned a different priority, in which the next job is taken from the highest-priority queue; within a queue, the system uses round-robin scheduling and a different time quanta depending on the priority of the queue. The time quanta increases as the priority decreases, varying the time allocated to a process when it runs. For this simulation, we use a simplified 3-queue configuration with the quantum for each queue set to  $2^i$ [18].

These are the five policies covered in our Operating Systems course and, thus, supported by our simulation. Additional policies are covered in other textbooks and could easily be added to GraySim.

## 4 Simulation

GraySim is written in Scala using a test-driven approach. We use the model-view-controller architectural pattern, as well as the observer and strategy patterns[4, 6]. Our testing platform is ScalaTest[19], a popular testing tool available in the Scala ecosystem. Our GUI is built using Swing[7].

GraySim is controlled and configured via the command line, in which students specify the policy (`--policy <policy>` where `<policy>` can be `fifo`, `mlfq`, `spn`, `stcf`, or `rr`), any parameters for the policy (e.g., `--quantum <q>` specifies the time quantum), as well as the start (aka arrival) time and service time of the processes that need to be scheduled (`--p <start> <service>`). GraySim supports all of the policies described in Section 3. Processes are named in the order they are specified beginning with ‘A’. A maximum of 10 processes can be scheduled, with no limit on length of service times. At least one process must start at time 0. The GraySim repository in GitHub contains a description of how to run the simulation in the README for users to reference (<https://github.com/usma-eecs/GraySim>).

Figure 1 shows the simulator upon start, showing a multi-level feedback queue policy. The process-related metadata is shown in the leftmost window of the screen, and we see five processes, named ‘A’ through ‘E’, that have start times ranging from time 0 through time 6 and service times ranging from 2 through 5 time units.

In the right-most area of the screen, three control buttons are available. The “Show Policy” button displays a written description of how the policy works at the top. The “Show Solution” button causes the solution to be visible and provides feedback to the student about any discrepancies between their solution and the correct solution. The feedback provided is specific to the student’s mistake(s). The “Configure” button causes the process and policy configuration to be visible.

In the center area of the screen, students can select which process is running during each time unit according to their understanding of the policy. Figure 2 shows the state of GraySim after the student has clicked on the “Show Policy” button in the right-hand control panel, has scheduled the processes according to their understanding of the multi-level feedback queue scheduling policy (where each queue,  $i$ , uses a quanta equal to  $2^i$ ) and has clicked on the “Show Solution” button. Their selections of the boxes are represented by a darkened cell in the scheduling table while the correct solution is represented by an ‘X’ in the cell. Figure 2 also shows the feedback window provided by GraySim to help the student correct their understanding. Figure 3 shows what the student will see once their solution is correct.

The feedback the student sees is one of two options: specific feedback for the current policy or generic feedback that applies to every policy. For example, First In First Out has a specific feedback message to inform students if they preempted a process because First In First Out is not a preemptive policy. The generic feedback that applies to every policy is double-booking processes, under-allocating or over-allocating time to a process, or starting a process before its determined start time. These feedback messages that appear to students when their solutions are not fully correct allow students to reflect on specific mistakes without needing to consult possibly inaccurate online resources or receive instructor feedback on their specific solution.

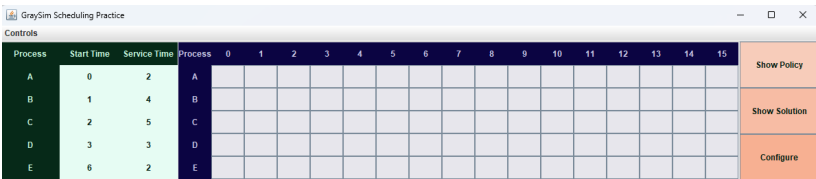


Figure 1: Overview of GraySim

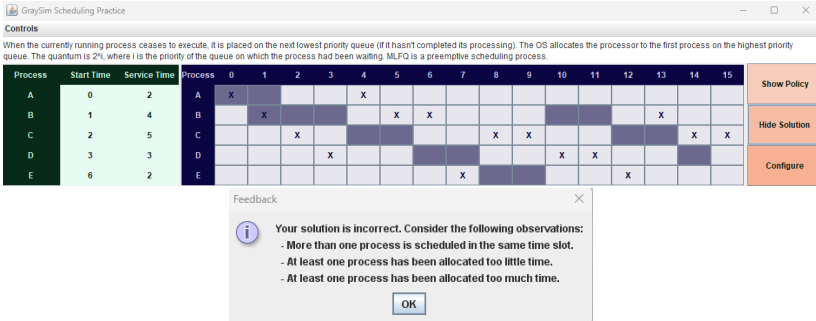


Figure 2: Incorrect Student Solution with Feedback

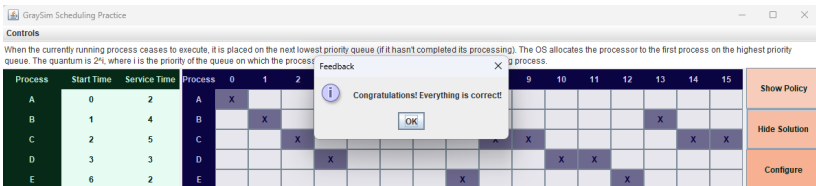


Figure 3: Correct Multi-Level Feedback Queue Solution

## 5 Evaluation

We performed a qualitative assessment of the scheduling simulation in November 2023 with the goal to gain an understanding of what current Operating Systems students thought of GraySim. The current students had learned these scheduling policies earlier in the semester, using the aforementioned paper-based worksheet. They were shown the nascent simulation and asked their perceptions of the tool for use in a future Operating Systems course.

### 5.1 Method

We asked students enrolled in Operating Systems to evaluate GraySim after getting a chance to observe GraySim via a demonstration. During the demonstration, they could request configuration changes (e.g., use a different policy, use 6 processes rather than 5, or change the start times or service times of the processes) and observe the results. If they wanted, the students were also given the opportunity to use GraySim themselves. After the demonstration was complete, they took a brief survey that asked the following questions:

1. If you had had access to this resource at the beginning of [the semester],

when you first learned about scheduling policies, how do you think it would have affected your understanding of the subject? (Five-point Likert scale from “Greatly improved” to “Greatly reduced”)

2. Which scheduling policies did you try? (Select any of first in first out, shortest job first, shortest time-to-completion first, round robin, and multi-level feedback queue)
3. Through using the simulation, how much, if at all, do you believe your current understanding of the scheduling policies you tried has changed? (Five-point Likert scale from “Greatly improved” to “Greatly reduced”)
4. What improvements could be made to the simulation? (Text entry)
5. What features do you like about the simulation that we should not lose sight of as we make modifications? (Text entry)
6. Did you encounter any bugs? (Select one of “yes”, “no”, and “maybe”; for those who responded with “yes” or “maybe”, text entry to describe the problem and how to reproduce it.)

All current students were invited to participate and anyone who did earned a total of 5 bonus points for their efforts. The total points for the course is 1000, so 5 bonus points represents a minimal incentive. Note that, by institutional policy, students who prefer not to participate can earn the same number of bonus points for an alternate activity; in this case, the alternate activity was a worksheet to practice their mastery of process scheduling policies.

Before the demonstration, the instructor checked the simulator’s responses for the configuration used in the demonstration to ensure that students were not shown incorrect results. In addition, the instructor also double-checked any configuration modifications requested by the students for the same reason.

## 5.2 Results

Of the 46 students enrolled in Operating Systems this semester, a total of 26 participated in the study. The students spent approximately 15-20 minutes viewing GraySim, experimenting with it, and answering the survey. The survey required an average of 7 minutes to complete with a minimum and maximum of 1.5 and 23.6 minutes respectively. Before analyzing the data, we removed any responses that were incomplete.

In response to the question about whether students believed that such a tool would have helped them learn the different OS scheduling policies if they had had access to it at the start of the semester, their responses were overwhelmingly positive with a few neutral responses and no negative responses, as

shown in Figure 4. We also asked students which of the OS scheduling policies they explored. They could select more than one policy option because many of them explored more than one during the study. Figure 5 shows their responses with multi-level feedback queue and shortest job first being the most commonly used policies during the study. We then asked students whether they believed that their current understanding of OS scheduling policies had increased due to their exposure to the simulation tool. As shown in Figure 6, their responses were less positive, with the vast majority responding that their understanding was somewhat improved or about the same.

The next two questions asked students what improvements they would like to see and what features they would like us to maintain. These were both open-ended text questions. In order to analyze them, one researcher created codes for the responses (e.g., allow process info to be randomized, change the way the solution is shown, etc.), and then both researchers used that coding to classify the responses. The researchers then compared their classifications and adjusted to achieve consensus.

Three comments provided in response to the question about which features the participants liked were actually comments that suggested improvements. The researchers agreed that these comments should be moved and analyzed with the other suggestions for improvement.

The students offered many suggestions for improvement, as shown in Table 1. The most common suggestion was to generate practice problems using randomized process start times and service times. The second-most requested features were to change how the student's solution and the correct solution are overlaid and to allow the solution to be shown incrementally so that students can test their understanding step-by-step.

Students also identified a number of features they found useful, as shown in Table 2. Our goal in requesting this feedback is to help us avoid losing sight of these features as we make improvements. The most popular feature is the ability to view the solution and to see feedback about which mistakes were made. Additionally, students liked the intuitive mechanism by which they identify their solution as well as their ability to see the policy.

## 6 Discussion & Future Work

Overall, we are encouraged by the results of this study. The overwhelming majority of students believe that the simulation will help students learn about scheduling policies for the first time. Although the current OS students did not feel their learning was greatly improved by the use of the simulation, we are not surprised or concerned by this result. These students were already familiar with the various scheduling policies and had already been tested on

Figure 4: Expected Learning with Access to Simulation At Start of Course

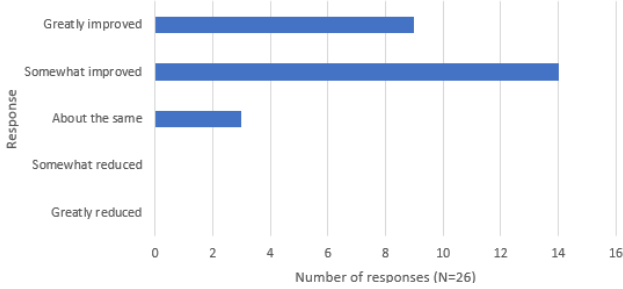


Figure 5: Scheduling Policies Explored

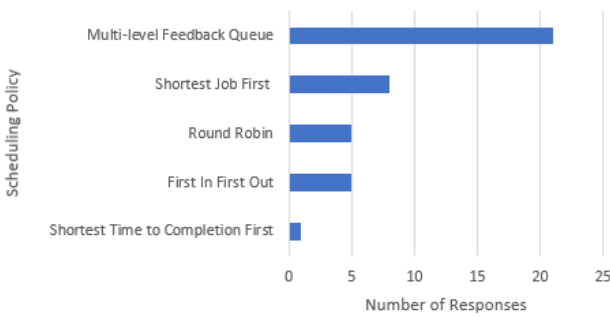


Figure 6: Incremental Learning with Access to Simulation at End of Course

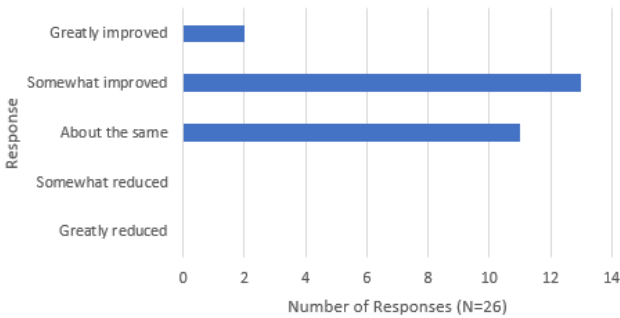


Table 1: Suggested Improvements

| <b>Feature</b>                                       | <b>Count</b> |
|--|--------------|
| Allow process information to be randomized           | 11           |
| Change the way the solution is overlaid              | 8            |
| Show solution incrementally                          | 8            |
| Allow process information to be specified in the app | 6            |
| Allow policy to be changed in the app                | 4            |
| Add a separate “Show Feedback” button                | 3            |
| Show the remaining service time                      | 2            |
| Show the state of the queues (where appropriate)     | 2            |
| Increase font sizes                                  | 2            |
| Display Policy Name                                  | 1            |
| Add grid around process information                  | 1            |
| Support timed competition                            | 1            |
| Give access via Course Website                       | 1            |

Table 2: Popular Features

| <b>Feature</b>                                | <b>Count</b> |
|---|--------------|
| Ability to view solution                      | 12           |
| Ability to see feedback                       | 7            |
| Mechanism to specify process schedule         | 6            |
| Ability to show policy                        | 6            |
| Ease of understanding interface               | 5            |
| Ability to configure process information      | 4            |
| Ability to configure policy                   | 3            |
| Recommend continuing paper-based ICE in class | 1            |
| Color scheme                                  | 1            |

them. Furthermore, they had little time with which to experiment with the simulation. Given the positive response to the simulation, we plan to continue our work during Spring 2024 to enhance GraySim with many of the requested changes. Our primary focus will be to add a feature to create a series of randomized processes for practice, to adjust the user interface, and to improve the feedback reported to students who answer incorrectly.

The top priority will be to support randomizing the process information. Although this feature seems like it would be trivial to add, it is not as easy as it sounds because care must be taken to ensure that the simulation is not overly long. We will need to include support for short-, medium-, and long-service times and ensure that the total amount of service time fits reasonably well on a laptop screen. The current process input option will also remain as a feature, so we need to integrate the two options in a user-friendly manner.

Another high priority will be to adjust the user interface to be more friendly, because the second-highest requested improvement is to change the way the solution is overlaid on the simulation screen. Currently, the solution is an 'X' or an ellipsis (when the screen is too small) written on top of the correct boxes. This interface is difficult to understand and see clearly, making it a necessary improvement for GraySim.

The final high priority item will be to improve the feedback provided to students when they answer incorrectly. Currently, GraySim supports some generic feedback that applies to all policies, such as a process has been allocated too much or too little time or a process was started before it arrived. Currently, only one of the policies provides policy-specific feedback. The First In, First Out policy detects if the student preempted a process and alerts them that the policy is non-preemptive. This feature was added as a proof-of-concept to help us evaluate its value. Given the popularity of this feature, additional policy-specific feedback needs to be identified and then a check needs to be implemented to add to the feedback feature when appropriate. In addition, we will add a separate button to allow students to see the feedback without seeing the solution. This will help students fix their answer before they see the exact solution.

We are also considering adding support for incorporating performance metrics, such as throughput and response time within GraySim (similar to [16]). These performance metrics would allow students to compare and contrast the performance of different policies given different workloads. The tool does not improve students' comparative knowledge right now because there is no way to view nor calculate performance metrics of the different policies. Adding support for such comparative analysis would be a valuable feature.

Our ultimate goal is to deploy this simulation to our Operating Systems course in the Fall 2024 semester. Given the positive feedback gathered in



this study, we expect students will have a positive learning experience with GraySim. We also hypothesize that the learning outcomes will improve because GraySim will allow for more practice and individualized feedback.

## 7 Conclusion

This paper presents a simulation that allows students enrolled in an undergraduate operating systems course to experiment with different CPU scheduling policies. The policies supported include First In First Out, Shortest Job First, Shortest Time-to-Completion First, Round Robin, and Multi-Level Feedback Queue. Students can specify a problem, either one that they generated or one their instructor provided, and test their understanding of the policy before requesting feedback from the simulation about the correctness of their solution.

We conducted a user study to evaluate the utility and usability of this scheduling simulator, inviting students currently enrolled in our local operating systems course to participate in the study. Participants provided overwhelmingly positive feedback on the utility of the scheduling simulator for use in our course. They also provided insights into key features of the tool and suggested numerous areas for enhancement.

Future work includes adding the ability to create randomized problems for practice, adjusting the user interface, and separating the feedback feature from the feature that shows the solution. The authors plan to continue working on GraySim in the coming semester, implementing the feedback obtained during the study, and improving the features offered. In Fall 2024, the authors plan to use GraySim in the Operating Systems Course and measure the quantitative change in learning outcomes. We hypothesize use of GraySim will improve student scores on the exam questions related to CPU scheduling.

## 8 Acknowledgements

The authors wish to thank the students who participated in the study for their valuable feedback and guidance as well as COL Christa Chewar for the inspiration and encouragement. The views expressed here are those of the authors and do not reflect the official policy or position of the Department of the Army, Department of Defense, or the U.S. Government.

## References

- [1] Mukul Agrawal. *CPU Scheduler Visualiser*. <https://github.com/mukul2310/cpu-scheduler-visualiser>. Accessed: 2023-12-11.

- [2] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. 1.10. Arpaci-Dusseau Books, Nov. 2023.
- [3] Peter DeRosa et al. “Realism and Simplicity: Disk Simulation for Instructional OS Performance Evaluation”. In: *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '06. Houston, Texas, USA: Association for Computing Machinery, 2006, pp. 308–312. ISBN: 1595932593. DOI: 10.1145/1121341.1121436. URL: <https://doi.org/10.1145/1121341.1121436>.
- [4] Object-Oriented Design. *Design Patterns*. <https://www.oodeesign.com/>. Accessed: 2023-12-6.
- [5] John Dickinson. “Operating Systems Projects Built on a Simple Hardware Simulator”. In: *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '00. Austin, Texas, USA: Association for Computing Machinery, 2000, pp. 320–324. ISBN: 1581132131. DOI: 10.1145/330908.331878. URL: <https://doi.org/10.1145/330908.331878>.
- [6] Erich Gamma et al. *Design Patterns*. Reading, MA, USA: Addison Welsey Longman, Inc, 1995.
- [7] Ben Hutchison and Hanns Holger Rutz (maintainers). *Scala Swing*. <https://index.scala-lang.org/scala/scala-swing>. Accessed: 2023-12-6.
- [8] Neetu Jain and P. V. Suresh. “A java based visual tool to learn CPU scheduling algorithms”. In: *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*. 2014, pp. 519–523. DOI: 10.1109/IndiaCom.2014.6828013.
- [9] Luiz Paulo Maia, Francis Berenger Machado, and Ageu C. Pacheco. “A Constructivist Framework for Operating Systems Education: A Pedagogic Proposal Using the SOsim”. In: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '05. Caparica, Portugal: Association for Computing Machinery, 2005, pp. 218–222. ISBN: 1595930248. DOI: 10.1145/1067445.1067505. URL: <https://doi.org/10.1145/1067445.1067505>.
- [10] Leo Natan Paschoal et al. “Towards an Educational Simulator to Support the CPU Scheduling Algorithm Education”. In: *2019 International Symposium on Computers in Education (SIIE)*. 2019, pp. 1–6. DOI: 10.1109/SIIE48397.2019.8970116.

- [11] Steven Robbins. “A Disk Head Scheduling Simulator”. In: *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '04. Norfolk, Virginia, USA: Association for Computing Machinery, 2004, pp. 325–329. ISBN: 1581137982. DOI: 10.1145/971300.971413. URL: <https://doi.org/10.1145/971300.971413>.
- [12] Steven Robbins. “A Three Pronged Approach to Teaching Undergraduate Operating Systems”. In: *SIGOPS Oper. Syst. Rev.* 42.6 (Oct. 2008), pp. 93–100. ISSN: 0163-5980. DOI: 10.1145/1453775.1453792. URL: <https://doi.org/10.1145/1453775.1453792>.
- [13] Steven Robbins. “A UNIX Concurrent I/O Simulator”. In: *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '06. Houston, Texas, USA: Association for Computing Machinery, 2006, pp. 303–307. ISBN: 1595932593. DOI: 10.1145/1121341.1121435. URL: <https://doi.org/10.1145/1121341.1121435>.
- [14] Steven Robbins. “An Address Translation Simulator”. In: *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '05. St. Louis, Missouri, USA: Association for Computing Machinery, 2005, pp. 515–519. ISBN: 1581139977. DOI: 10.1145/1047344.1047507. URL: <https://doi.org/10.1145/1047344.1047507>.
- [15] Steven Robbins. “Experimentation with Bounded Buffer Synchronization”. In: *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '00. Austin, Texas, USA: Association for Computing Machinery, 2000, pp. 330–334. ISBN: 1581132131. DOI: 10.1145/330908.331880. URL: <https://doi.org/10.1145/330908.331880>.
- [16] Steven Robbins. “Introducing Empirical Investigation in Undergraduate Operating Systems”. In: *SIGOPS Oper. Syst. Rev.* 32.4 (Oct. 1998), pp. 77–80. ISSN: 0163-5980. DOI: 10.1145/302350.302366. URL: <https://doi.org/10.1145/302350.302366>.
- [17] Steven Robbins. “Starving Philosophers: Experimentation with Monitor Synchronization”. In: *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '01. Charlotte, North Carolina, USA: Association for Computing Machinery, 2001, pp. 317–321. ISBN: 1581133294. DOI: 10.1145/364447.364612. URL: <https://doi.org/10.1145/364447.364612>.
- [18] William Stallings. *Operating Systems: Internals and Design Principles*. 9th. USA: Prentice Hall Press, 2011. ISBN: 0-13-467095-7.
- [19] Bill Venners and Artima. *ScalaTest*. <https://www.scalatest.org/>. Accessed: 2023-12-6.

# A Pleasant Surprise: A Classic Assignment and an Offbeat Assessment for PDC, HPC, and related KU Coverage\*

Nicholas S. Rosasco, Andrew Paxson, Ethan Hawk  
Computing and Information Sciences  
Valparaiso University

## Abstract

This paper presents an assignment that when paired with a discussion or other follow-up can be used to increase and diversify computer science knowledge unit coverage. The knowledge units are taken from the CS2013 and related documents from ACM/IEEE. This guidance to the teaching community notes the criticality of educating students on parallel, distributed, and high performance computing. Information on available, modular resources for addressing these areas in the classroom is presented. Additionally, integrating undergraduate writing expectations into a topics course is discussed.

## 1 Introduction

This paper is an exercise in documentation, presenting post-hoc assessment at a granular level of an assignment that has been used twice for teaching a spread of parallel and distributed and high performance computing (PDC, HPC) topics. Additionally, it provides a demonstration of how coverage of topics can be easily increased through a single assignment - even one not designed for assessment use. It uses one of the key pieces of current curricular guidance for Computer Science, CS2013[1], which openly states that it isn't possible to cover all of the topics and areas it lists. This reality makes creating and sharing any tools

---

\*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.

that provide greater coverage of some import for the wider community, and ultimately motivates this paper.

## 2 Background and Context

The original use of this assignment was at Valparaiso University, which has a Computer Science major that has been configured for maximum flexibility - essentially just four required classes plus math requirements and a capstone. With this flexibility comes a risk of omission due to the limited time available. Specifically, there is a very real risk of omission due to limited time with required courses. Given those risks and the CS2013 and related guidance, it became evident that there was a gap in parallel and distributed computing content coverage. Accordingly, a series of topics classes has been run to provide that knowledge and experience. Two of those classes have used the assignment being presented. Further, Valparaiso University has a strong liberal arts tradition that includes an emphasis on undergraduate writing which also informed the configuration of the classes, particularly with regard to this assignment. As the CS curriculum at Valparaiso University has limited research-driven writing, this assignment became a way to provide more practice with academic and technical writing. This also fit well with recent updates to the Undergraduate Writing Program at Valparaiso - and complements recent efforts to introduce the history of computing into the classroom. While the perhaps obligatory grumble about writing was heard, the use of history inspired students to engage with the assignment. In the second iteration of the assignment, a presentation component was added that also required a recorded rehearsal to reinforce the communications aspects of the assignment.

The assignment was designed to fit into an HPC topics course, which would normally be largely driven by introducing technologies. The running of topics classes is also a regular component of Valparaiso's efforts to build a highly flexible major. The class was largely built around modules, sourced from the impressive resources listed in Prior Work, that provided a hands-on, lab-focused backbone. The inclusion of a classic research-paper style task was in part meant to broaden the reach of information provided by pushing students to use books and academic papers, complementing the institutional efforts on information literacy. The students, to reinforce this, would have slightly over a month to tackle the writing. [It should be noted that the class was last run just before any generative AI tools became a significant pedagogical concern.] This time-frame was also meant to encourage use of the institutional library and specialist staff available there; the ACM Digital Library is particularly emphasized as a resource.

The assignments provided a writing prompt to students along with a list of

historic super-computing systems (full list at <http://www.valpo.edu/computing-information-sciences/files/2024/02/Historical-List-of-Supercomputers.pdf>; see also the first figure for a subset), which is intended to leverage the history of the high performance systems community as raw material.

| Name of Computer | When it was first built | Who made it (Company and Designer)                                 | Built for/Used by... | Notes/ Important Details   | Useful source(s) on it   |
|------------------|-------------------------|--|----------------------|--|--|
| Illiac IV        | 1971                    | UIUC, DARPA, Burroughs, Texas Instruments (Chief: Daniel Slotnick) | NASA - Ames Lab      | First supercomputer on arpanet; early massively parallel design  | 'The Illiac IV System', WJ Bouknight et al, Proceedings of the IEEE Vol 60, No. 4, April 1972                  |
| Cray 1           | 1975                    | Cray Research / Seymour Cray                                       | Multiple customers   | One of the first 'production' supercomputers                     |  |
| Gordon           | January, 2012           | San Diego Supercomputer center                                     |                      | It will help for research of further Large Hadron Collider (LHC) | "SDSC's Gordon Supercomputer Assists in Crunching Large Hadron Collider Data." Energy Weekly News (2013): 422. |

Figure 1: Historic Computing Systems - Subset/Example

The prompt included the following questions; information in brackets was provided as additional guidance:

1. How does the system fit into the bigger picture? [Options here include talking about what problems was it built to solve, what was available technology wise, what limits / constraints informed the plan and building, etc.]
2. Describe the system [vector? massively parallel? liquid cooled? etc] in terms of architecture, chips, the usual trinity (power space cooling, what power it was meant to deliver and what it did deliver, etc. Some demonstration of comprehension of this would be useful; a definition may be good along with a comparison to a current system or two.
3. What sort of advance was it? [Why are we talking about it?]
4. How did it fit (or not) the job it was built for - or how was the job adjusted for it. [Tell the bigger story.]

The hope was that, after completing the assignment, students would have a greater appreciation that the HPC systems of the past have proven to be the prologue for everyday computing.

### 3 Prior Work

The course design for the assignment presented here was informed by a variety of projects and efforts. While CS2013 was itself the original motivation for the class, a variety of inputs from the pedagogic-focused community and the campus context became factors.

#### 3.1 Initial Motivation and Materials

CS2013 places significant emphasis on the area of parallel and distributed computing (PDC). The overhead [4] of mastering the infrastructure and systems also create significant barriers in teaching in this area. More generally, the challenges of teaching and using HPC/PDC technologies remain an ongoing discussion and concern, see [15], [31] and [33]. Happily, significant and high quality efforts have been made to provide resources for covering this area. First and foremost, the high quality modules built by the CSinParallel team and project provide a convenient and engaging foundation. These can be mixed and matched, and are a rich and excellent suite of options for coding challenges and labs; the team and contributors have also various introductory and training events for using their materials. The contributors, community, and workshops hosted by this group were critical assets [35],[26],[6],[7],[27],[8],[2],[9],[11],[10],[20],[25],[12],[13]. The work done by others was also helpful [14],[21],[17], particularly as it similarly used a modular approach. The iPDC work and related efforts were also notable [30],[19]. Specific mention should also be made of the work of David Toth, particularly with the ODROID platforms [3]. His early help and enthusiasm at a couple of key moments was particularly impactful as was the information published by Charlie Peck and others on the LittleFe clusters [29],[28] and the linked Bootable Linux Cluster CD project [24]. Other recent, related work done in a modular style includes the ToUCH Project at which nicely extends the work done for traditional HPC/PDC with excellent additional modules on heterogeneous computing [32]. It should be noted that the lab-centered approaches - woven through the hands-on exercises mentioned above - are critical. While these resources proved profoundly useful in constructing the classes and helping frame the lectures, some mechanisms were needed. This need provided the bridge to the next input - an ongoing effort on disciplinary writing.

#### 3.2 Undergraduate writing

Valparaiso University invests heavily in its Undergraduate Writing Program. One of the touchstones for that effort was the book *Engaging Ideas: The Professor's Guide to Integrating Writing, Critical Thinking, and Active Learning*

in the Classroom [5]. It presents writing as a means to creating understanding. Since the lab assignments necessarily were group work, so as to share hardware, a writing assignment provided a way to individualize work for students, which was also helpful. Since a key theme of the course involved the transition of HPC/PDC concepts from the most complex systems to the the consumer level, some further emphasis on that pattern was also desired. A per-person selection of a legacy system provided a way to also create distinctly individual effort, as most other effort in the course happened as group assignments.

### 3.3 Systems thinking

The centrality of the hardware/software combination as key to PDC/HPC impact (typically speedup) was a theme of the course. To underpin this, and to illustrate the rapidity of evolution, occasional pauses to look at historical approaches to dealing the more engineering-derived constraints were built into the course. While many students preferred a focus on algorithms and code (the phrase "icky hardware" was sometimes heard), the explicit use of the history of computing was a departure from the usual and inspired by presentations like the SIGCSE sessions like 'Computer History on the Move'[16]. Another recurring idea for the course, was the dominant trio of factors for HPC solutions: power, space, and cooling as constraints. The need to balance the cost and operating constraints that directly or indirectly frame the issues raised by that somewhat unholy (?) trinity was a recurring theme of the class. Including that recurring problem as a concept, even without a codified KU, within the assessment setup for the assignment was motivated by the linkage to the in-class discussions; in effect, it serves as a baseline measure.

### 3.4 The CS2013 Knowledge Units

CS2013's curricular survey makes explicit the need for this area to be addressed [1] and provides a highly granular list - the KUs. The initial look at the student work in the normal process of grading indicated that a surprising number of KUs were (or could be) covered or were at least introduced by the assignment - and somewhat unexpectedly! That opened the door to follow up questions especially given work like [9]. These included:

- How many things got covered in this one exercise?
- How useful was this one additional assignment, when stacked up against CS2013's identifications? With that thought, a close look at CS2013 was in order - with the student responses in hand. This, in turn led to the traditional assessment concern - how well and how much was addressed.

Accordingly, the CS2013 KU list was reviewed alongside both the assignment prompt and the submissions. While the size and granularity of CS2013 makes



navigating the intellectual terrain somewhat akin to refolding a large highway map, a set of KUs were selected to show both some spread and the first impression of the student efforts.

### 3.5 Related Tools

ACM's Computing Curriculum 2020 also notes the importance of this topic area [18]. The drafts and discussion of CS2023 indicate the ongoing emphasis on parallel. The linkage to the exploding area of AI, in particular, also emphasized the ongoing importance of this topic [23], [34], [22].

## 4 Assessment Process

For this discussion, the following knowledge units from the AR, CN, OS, and PD areas of CS2013 were considered, these are incorporated into the figure below. In addition, a faculty defined point, not separately identified in 2013,

1. (Digital logic and digital systems) Comprehend the trend of modern computer architectures towards multi-core and that parallelism is inherent in all hardware systems.
2. (Digital logic and digital systems) Explain the implications of the "power wall" in terms of further processor performance improvements and the drive towards harnessing parallelism.
3. (Assembly level machine organization) Describe instruction level parallelism and hazards, and how they are managed in typical processor pipelines.
4. (Multi-processing and alternative architectures) Discuss the concept of parallel processing beyond the classical von Neumann model.
5. (Multi-processing and alternative architectures) Describe alternative parallel architectures such as SIMD and MIMD.
6. (Processing) Describe the levels of parallelism including task, data, and event parallelism.
7. (Device Management) Explain the key difference between serial and parallel devices and identify the conditions in which each is appropriate.
8. (Parallelism Fundamentals) Distinguish using computational resources for a faster answer from managing efficient access to a shared resource.
9. (Parallel Algorithms, Analysis, and Programming) Characterize features of a workload that allow or prevent it from being naturally parallelized.
10. (Parallel Architecture) Explain the differences between shared and distributed memory.
11. (Parallel Architecture) Describe the SMP architecture and note its key features.
12. (Parallel Architecture) Characterize the kinds of tasks that are a natural match for SIMD machines.
13. (Parallel Architecture) Describe the advantages and limitations of GPUs vs CPUs.
14. (Parallel Architecture) Explain the features of each classification in Flynn's taxonomy.
15. (Parallel Architecture) Describe the key performance challenges in different memory and distributed system topologies.
16. (Parallel Performance) Explain the impact and trade-off related to power usage on parallel performance.
17. (Assembly level machine organization) Explain different instruction formats, such as addresses per instruction and variable length vs fixed length formats.
18. (Memory system organization and architecture) Describe how the use of memory hierarchy (Cache, virtual memory) is used to reduce the effective memory latency.
19. (Interfacing and communication) Identify various types of buses in a computer system.
20. (Interfacing and communication) Compare common network organizations, such as ethernet/bus, ring, switched vs routed.

Figure 2: Knowledge Units Selected for Consideration

was also included based on the history of HPC systems, which also touched on a recurring theme in the course and could be used as a baseline: Note the importance of cooling.

To assess these, an approach in some ways similar to traditional grading was adopted. The second and third authors, each with extensive experience as undergraduate teaching assistants, scored the writing assignment submissions against the numbered items above. This scoring done in two ways, as appropriate to the KU, detailed in the figures below: The results of that scoring can be illustrated with simple graphs shown in the following figures; note the charts are grouped by scoring style. These results show varying levels of im-

1. Vague or indistinct discussion of item / "gets near the idea"
2. Indirect reference or context makes it apparent that it was considered
3. Noted or referenced/mention was made of concept
4. Some explicit discussion
5. Well informed comment or discussion, possibly at some length

Figure 3: Possible scores - Likert

- Yes, the idea/topic appears;
- No means not discussed or discussed incorrectly;
- Maybe means there is a vague reference or adjacent idea, but actual idea use is uncertain
- NA is not applicable (it isn't relevant/wouldn't necessarily appear in this submission)

Figure 4: Possible scores - "Bins"

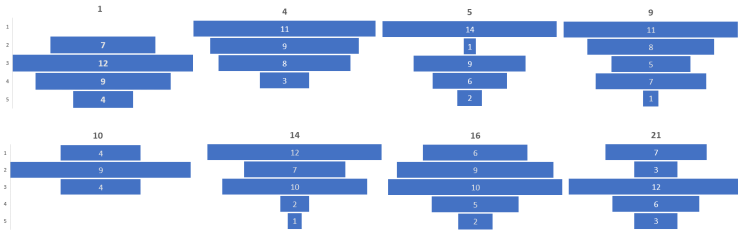


Figure 5: Charts of 1..5 Scores - Number of KU above each funnel block-set

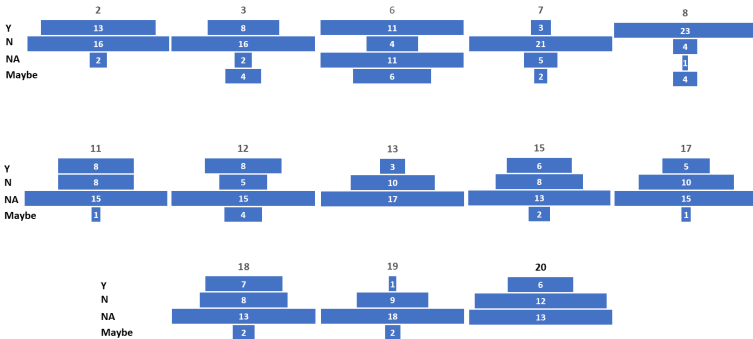


Figure 6: Charts of Y/N/NA/Maybe Scores - Number of KU above each funnel block-set

pact - but also demonstrate that a broad engagement with the KUs (and the supplemental item) was achieved. This is somewhat compelling when considered against the likely coverage - perhaps only a bullet on a slide - that would occur this assignment. This intriguing spread of coverage was reinforced - and spread across all students - by a general class discussion in the first edition of the course; it was later supplemented with a presentation component as previously mentioned - as a deliberate leverage of the discussion in the first version. While admittedly somewhat subjective and spread out across multiple areas, this set of student results and scores is broadly similar to assessment practices used across both US-regional accrediting bodies and ABET-participating institutions. It also, more broadly, shows the serendipitous effect of this style of assignment.

## 5 Conclusions and Future Work

If repeating this assignment, a peer revision step would be added for the writing assignment. The in-class presentation component (and/or recorded delivery) will also continue to be component. In the case of Valparaiso University, a mix of majors was represented; discussion and questions for both semesters this course was run benefited from the different mindset and approaches. Peer review, if used next time, will would be done to encourage crossing disciplinary lines, with Computer Science and Computer Engineering students paired when swapping their drafts to increase the diversity of thought. In subsequent efforts, some sort of incentive mechanism would also be implemented to generate a bit more engagement and thought around choosing a platform. This might be as simple as a preliminary paragraph or two demonstrating interest or some background investigation. In all, this assignment provided a somewhat pleasant surprise. The in-class engagement increased and the range of KUs touched on - while perhaps not especially consistent on depth - is somewhat striking. This consumed, effectively, a class period and a half counting the introduction of the assignment, some follow up questions and the day of discussion. The amount of topical coverage, for such a small class time investment, made this very traditional style of assignment a pleasant surprise.

## 6 Acknowledgements

Thanks to Michael Paul Clemens, of the University of Utah, for varied and valuable feedback, as well as to the Department and students of Computing and Information Sciences at Valparaiso University for the support they provided. Much appreciation for the CSinParallel community, ToUCH, and related modular efforts should be noted as well for without their work, not to men-

tion their warm welcome and support, time to pursue alternative assignments beyond (re)creating customary lab exercises would have been unavailable.

## References

- [1] ACM Computing Curricula Task Force, ed. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, Inc, Jan. 2013. ISBN: 9781450323093. DOI: 10.1145/2534860. URL: <http://dl.acm.org/citation.cfm?id=2534860>.
- [2] Joel C. Adams and Elizabeth Shoop. “Teaching Parallel & Distributed Computing with MPI \* Tutorial Presentation”. In: *Journal of Computing Sciences in Colleges* 33.1 (2017). DOI: <https://dl.acm.org/doi/10.5555/3144605.3144626>.
- [3] Joel C. Adams et al. “The Micro-Cluster Showcase”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: ACM, Feb. 2016, pp. 82–83. ISBN: 9781450336857. DOI: 10.1145/2839509.2844670. URL: <https://dl.acm.org/doi/10.1145/2839509.2844670>.
- [4] Ivan Babic et al. “LittleFe and BCCD as a successful on-ramp to HPC”. In: *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. New York, NY, USA: ACM, July 2014, pp. 1–7. ISBN: 9781450328937. DOI: 10.1145/2616498.2616569. URL: <https://dl.acm.org/doi/10.1145/2616498.2616569>.
- [5] John C. Bean. *Engaging Ideas: The Professor’s Guide to Integrating Writing, Critical Thinking, and Active Learning in the Classroom*. 2e. Jossey-Bass, 2011. ISBN: 0470532904.
- [6] Richard Brown et al. “Teaching Parallel and Distributed Computing with MPI on Raspberry Pi Clusters”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2018, pp. 1054–1054. ISBN: 9781450351034. DOI: 10.1145/3159450.3162369. URL: <https://dl.acm.org/doi/10.1145/3159450.3162369>.
- [7] Richard A. Brown and Edward F. Gehringer. “Developing competency in parallelism”. In: *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*. New York, NY, USA: ACM, Oct. 2012, pp. 229–230. ISBN: 9781450315630. DOI: 10.1145/2384716.2384783. URL: <https://dl.acm.org/doi/10.1145/2384716.2384783>.

- [8] Richard A. Brown, Elizabeth Shoop, and Joel Adams. “CSinParallel using map-reduce to teach parallel programming concepts across the CS curriculum”. In: *Proceeding of the 44th ACM technical symposium on Computer science education*. New York, NY, USA: ACM, Mar. 2013, pp. 758–758. ISBN: 9781450318686. DOI: 10.1145/2445196.2445504. URL: <https://dl.acm.org/doi/10.1145/2445196.2445504>.
- [9] Richard A. Brown, Elizabeth Shoop, and Joel C. Adams. “Bringing CS2013 recommendations for parallel and distributed computing into your CS curriculum”. In: *Proceedings of the 45th ACM technical symposium on Computer science education*. New York, NY, USA: ACM, Mar. 2014, pp. 79–80. ISBN: 9781450326056. DOI: 10.1145/2538862.2538996. URL: <https://dl.acm.org/doi/10.1145/2538862.2538996>.
- [10] Richard A. Brown, Elizabeth Shoop, and Joel C. Adams. “CSinParallel Using WebMapReduce to Teach Parallel Computing Concepts, Hands-on”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: ACM, Feb. 2016, pp. 716–717. ISBN: 9781450336857. DOI: 10.1145/2839509.2844688. URL: <https://dl.acm.org/doi/10.1145/2839509.2844688>.
- [11] Richard A. Brown et al. “A stratified view of programming language parallelism for undergraduate CS education”. In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2012, pp. 81–82. ISBN: 9781450310987. DOI: 10.1145/2157136.2157162. URL: <https://dl.acm.org/doi/10.1145/2157136.2157162>.
- [12] Richard A. Brown et al. “Strategies for adding the emerging PDC curriculum recommendations into CS courses”. In: *Proceeding of the 44th ACM technical symposium on Computer science education*. New York, NY, USA: ACM, Mar. 2013, pp. 109–110. ISBN: 9781450318686. DOI: 10.1145/2445196.2445232. URL: <https://dl.acm.org/doi/10.1145/2445196.2445232>.
- [13] Richard A. Brown et al. “Teaching parallel design patterns to undergraduates in computer science”. In: *Proceedings of the 45th ACM technical symposium on Computer science education*. New York, NY, USA: ACM, Mar. 2014, pp. 547–548. ISBN: 9781450326056. DOI: 10.1145/2538862.2538875. URL: <https://dl.acm.org/doi/10.1145/2538862.2538875>.
- [14] Martin Burtscher et al. “A Module-based Approach to Adopting the 2013 ACM Curricular Recommendations on Parallel Computing”. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2015, pp. 36–41. ISBN:

9781450329668. DOI: 10.1145/2676723.2677270. URL: <https://dl.acm.org/doi/10.1145/2676723.2677270>.
- [15] Juan Chen, Sheikh Ghafoor, and John Impagliazzo. “Producing competent HPC graduates”. In: *Communications of the ACM* 65.12 (Dec. 2022), pp. 56–65. ISSN: 0001-0782. DOI: 10.1145/3538878. URL: <https://dl.acm.org/doi/10.1145/3538878>.
- [16] Graham Farr et al. “Computer History on the Move”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. New York, NY, USA: ACM, Feb. 2016, pp. 528–533. ISBN: 9781450336857. DOI: 10.1145/2839509.2844575. URL: <https://dl.acm.org/doi/10.1145/2839509.2844575>.
- [17] Andrew Fitz Gibbon et al. “Teaching high performance computing to undergraduate faculty and undergraduate students”. In: *Proceedings of the 2010 TeraGrid Conference*. New York, NY, USA: ACM, Aug. 2010, pp. 1–7. ISBN: 9781605588186. DOI: 10.1145/1838574.1838581. URL: <https://dl.acm.org/doi/10.1145/1838574.1838581>.
- [18] CC2020 Task Force. *Computing Curricula 2020: Paradigms for Global Computing Education*. ACM, Nov. 2020. ISBN: 9781450390590. DOI: 10.1145/3467967. URL: <http://dx.doi.org/10.1145/3467967>.
- [19] Sheikh Ghafoor, Sushil Prasad, and Charles Weems. “Integrating Parallel and Distributed Computing in Early CS Courses”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. New York, NY, USA: ACM, Mar. 2022, pp. 1198–1198. ISBN: 9781450390712. DOI: 10.1145/3478432.3499155. URL: <https://dl.acm.org/doi/10.1145/3478432.3499155>.
- [20] Sheikh K. Ghafoor et al. “Unplugged Activities to Introduce Parallel Computing in Introductory Programming Classes”. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, July 2019, pp. 309–309. ISBN: 9781450368957. DOI: 10.1145/3304221.3325573. URL: <https://dl.acm.org/doi/10.1145/3304221.3325573>.
- [21] David A. Joiner et al. “Teaching parallel computing to science faculty”. In: *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*. New York, NY, USA: ACM, Mar. 2006, pp. 239–246. ISBN: 1595931899. DOI: 10.1145/1122971.1123007. URL: <https://dl.acm.org/doi/10.1145/1122971.1123007>.

- [22] Amruth N. Kumar and Rajendra K. Raj. “A First Look at the ACM/IEEE-CS/AAAI Computer Science Curricula (CS202X)”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. New York, NY, USA: ACM, Mar. 2022, pp. 1023–1024. ISBN: 9781450390712. DOI: 10.1145/3478432.3499036. URL: <https://dl.acm.org/doi/10.1145/3478432.3499036>.
- [23] Amruth N. Kumar and Rajendra K. Raj. “Computer Science Curricula 2023 (CS2023)”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*. New York, NY, USA: ACM, Mar. 2022, pp. 1212–1213. ISBN: 9781450394338. DOI: 10.1145/3545947.3569591. URL: <https://dl.acm.org/doi/10.1145/3545947.3569591>.
- [24] Baochuan Lu. “Use bootable Linux CD (BCCD) to teach cluster and parallel computing concepts: conference workshop”. In: *Journal of Computing Sciences in Colleges* 24.5 (2009). DOI: <https://dl.acm.org/doi/10.5555/1516595.1516625>.
- [25] Suzanne J. Matthews. “Teaching with Parallella: A First Look in an Undergraduate Parallel Computing Course”. In: *Journal of Computing Sciences in Colleges* (2015). DOI: <https://dl.acm.org/doi/10.5555/2835377.2835381>.
- [26] Suzanne J. Matthews et al. “Portable Parallel Computing with the Raspberry Pi”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2018, pp. 92–97. ISBN: 9781450351034. DOI: 10.1145/3159450.3159558. URL: <https://dl.acm.org/doi/10.1145/3159450.3159558>.
- [27] Suzanne J. Matthews et al. “Teaching Parallel Computing with OpenMP on the Raspberry Pi (Abstract Only)”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, Mar. 2017, pp. 741–741. ISBN: 9781450346986. DOI: 10.1145/3017680.3017818. URL: <https://dl.acm.org/doi/10.1145/3017680.3017818>.
- [28] Tim McGuire. “Using LittleFe to introduce parallel and distributed computing in the curriculum”. In: *Journal of Computing Sciences in Colleges* 29.4 (2014), p. 50. DOI: <https://dl.acm.org/doi/10.5555/2591468.2591477>.
- [29] Charles Peck et al. “LittleFe buildout workshop (parts 1 & 2)”. In: *Proceeding of the 44th ACM technical symposium on Computer science education*. New York, NY, USA: ACM, Mar. 2013, pp. 763–763. ISBN: 9781450318686. DOI: 10.1145/2445196.2445520. URL: <https://dl.acm.org/doi/10.1145/2445196.2445520>.

- [30] Sushil Prasad et al. “Modernizing Early CS Courses with Parallel and Distributed Computing”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2019, pp. 1241–1241. ISBN: 9781450358903. DOI: 10.1145/3287324.3287548. URL: <https://dl.acm.org/doi/10.1145/3287324.3287548>.
- [31] Sushil Prasad et al. “NSF/IEEE-TCPP Curriculum on Parallel and Distributed Computing for Undergraduates - Version II - Big Data, Energy, and Distributed Computing”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*. New York, NY, USA: ACM, Mar. 2022, pp. 1220–1221. ISBN: 9781450394338. DOI: 10.1145/3545947.3569594. URL: <https://dl.acm.org/doi/10.1145/3545947.3569594>.
- [32] Apan Qasem and David Bunde. “Heterogeneous Computing for Undergraduates”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. New York, NY, USA: ACM, Mar. 2022, pp. 1201–1201. ISBN: 9781450390712. DOI: 10.1145/3478432.3499152. URL: <https://dl.acm.org/doi/10.1145/3478432.3499152>.
- [33] Rajendra K. Raj et al. “High Performance Computing Education”. In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, June 2020, pp. 51–74. ISBN: 9781450382939. DOI: 10.1145/3437800.3439203. URL: <https://dl.acm.org/doi/10.1145/3437800.3439203>.
- [34] Rajendra K. Raj et al. “Perspectives on Computer Science Curricula 2023 (CS2023)”. In: *Proceedings of the ACM Conference on Global Computing Education Vol 2*. New York, NY, USA: ACM, Dec. 2023, pp. 187–188. ISBN: 9798400703744. DOI: 10.1145/3617650.3624928. URL: <https://dl.acm.org/doi/10.1145/3617650.3624928>.
- [35] Elizabeth Shoop et al. “Virtual clusters for parallel and distributed education”. In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. New York, NY, USA: ACM, Feb. 2012, pp. 517–522. ISBN: 9781450310987. DOI: 10.1145/2157136.2157287. URL: <https://dl.acm.org/doi/10.1145/2157136.2157287>.



# Evaluating the Efficacy of Peer-Created Worked-Example Videos in a Computer Systems Course\*

Grace Kim, Dylan Green, and Suzanne J. Matthews  
Department of Electrical Engineering & Computer Science  
United States Military Academy  
West Point, NY 10996

[[grace.kim](mailto:grace.kim), [dylan.green](mailto:dylan.green), [suzanne.matthews](mailto:suzanne.matthews)][@westpoint.edu](mailto:westpoint.edu)

## Abstract

Worked examples are an educational tool widely used in introductory computer science classes, primarily for programming and code-tracing concepts. Prior research supports the use of worked examples as a scaffolding mechanism to help students build a solid foundation before tackling problems on their own. Whether breaking down the intricacies of code or explaining abstract theoretical concepts, worked examples offer a structured approach that nurtures a deeper understanding during self-study. This study explores how *peer-created* worked examples, shown through detailed step-by-step videos, aid student learning in an intermediate-level computer science course, namely computer systems.

Our results suggest that worked-example videos are a useful study aid for intermediate computer science courses, such as computer systems. Students who watched the worked-example videos found them to be very helpful, and ranked them as the top study aid for succeeding on quizzes. Additionally, students with access to worked-example videos performed moderately better on quizzes compared to students without worked-example videos. Our results and experiences also suggest that worked-example videos are beneficial to the students who created them as well as their peers who use them.

---

\*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

Worked examples (also known as worked-out examples or worked solutions) are a pedagogical technique widely employed in teaching computer science. Formally, worked examples contain some formulation of a problem, some information on how to derive the solution, and the final answer [1]. Worked examples bridge the gap between the task and the answer, providing a solid and accurate foundation for students to later practice solving problems on their own [16, 10]. Worked examples have been widely cited as effective tools in the initial stages of learning procedural concepts in a wide variety of subjects[10], including algebra[14], chemistry[7], and english[5]. Students relied on these worked solutions to practice problems while reassuring themselves that they understood the necessary skills and underlying concepts [3].

Worked examples come in many forms, including text-based static examples, such as solution explanations in textbooks that are presented statically and all at once; or modeling examples, in which a teacher or peer generates a solution in real-time, allowing learners to see the solution built step by step; or dynamic examples, in which a custom tool, software, or animation presents a step-by-step solution of a problem of a code trace either using a custom tool or through an animation [9].

In the field of computer science, researchers have primarily delved into the impact of worked examples on introductory programming. As such, the efficacy of worked examples that illustrate code-tracing examples and program-building are widely studied [9, 12]. Skudder [12] describes worked examples in computer science as a “signature pedagogy”; however, most research on worked examples focus on instructor-created worked examples [8, 13, 6, 15, 4]. Prior work suggests that while students appreciate worked example videos containing code demonstrations, there was no statistically significant effect on student learning.

Research also suggests that this phenomenon is not restricted to worked examples that cover programming-only concepts. A recent study by Zavgorodniaia et. al [15] studied the effect of worked-example videos that diagrammatically explained Dijkstra’s algorithm on a population of undergraduates who were primarily non-majors. The researchers found that access to the videos did not have a statistically significant effect on student learning, supporting an earlier result by Morrison [8].

Another recent study [4] performed a qualitative analysis on student perceptions of instructor-created video recordings of lectures, in which instructors presented static examples and live coding examples to students in an introductory computer systems course, and surveyed students on their thoughts on the two techniques. We note students had no control over the pacing of the videos, and could not pause them once started. The researchers reported that

students found value in both modalities, liking the “at their own pace” studying and “focus on finished product” enabled by static worked examples, but the insights on instructor reasoning and “development process” of the live coding. In all the aforementioned cases, the videos were created by instructors, largely to control for high quality.

This paper looks at the impact of worked-example videos in a computer systems course at West Point, a four-year baccalaureate college. Our work is novel for several reasons. First, we evaluate *peer-created* worked example videos on student performance and perceptions, rather than instructor-created videos. Second, in addition to standard worked examples of program building and tracing, a non-trivial number of the produced worked-example videos involve non-programming content, such as reverse engineering (where learners observe how an assembly program translates to C code), cache address mapping, and visualizing process execution. Thus, our work adds to the body of knowledge on the effectiveness of worked example videos for non-programming content.

The rest of the paper is organized as follows. Section 2 provides an overview of our methodology, including details on the course implementation the worked-example video creation process, and experimental setup. Section 3 discusses the results of our quantitative and qualitative analyses. Lastly, we offer some reflections on lessons learned and major conclusions in Section 4.

## 2 Methodology

Data was collected over two fall offerings of CS380, a computer systems course taken at West Point, typically during junior year. The course generally represents students’ first exposure to computer systems topics and is required for all computing majors at West Point. CS380 is split up into four units: C, Assembly, Memory & Code Optimization, and Parallel Computing. Concepts discussed in class are primarily evaluated through ten quizzes that are spread out over the semester, with one to three quizzes given every unit. The remainder of the course grade is determined by a series of multi-week projects, designed to build students’ programming and assembly reading skills.

The investigators of this study include two undergraduate students and a faculty advisor. The undergraduate investigators took CS380 during Fall 2022 during their junior year, alongside their classmates. While enrolled in the course, they independently worked with the faculty advisor to create worked-example videos. The following semester, the students completed IRB training and worked with their advisor to design the study, which was reviewed and authorized by West Point’s IRB process; the full study was executed in Fall 2023. The two students received independent study credit for the semesters they worked on the project.

Table 1 shows the spread of quizzes, in-class-exercises (ICEs) and corresponding worked-example videos produced throughout all of the units in CS380. Nearly half (42.5%) of the videos were created for the C unit, partially because this is the unit that students have traditionally had the greatest amount of difficulty in the course, and because it contains the largest number of corresponding in-class exercises.

Table 1: Number of Worked-example Videos in each Unit

| Unit Number and Name       | Num. Lessons | Num. Quizzes | Num. Videos | Num. ICEs |
|----------------------------|--------------|--------------|-------------|-----------|
| Unit 1: C                  | 9            | 2            | 17          | 20        |
| Unit 2: Assembly           | 10           | 3            | 9           | 14        |
| Unit 3: Memory & Code Opt. | 9            | 2.5          | 8           | 17        |
| Unit 4: Concurrency        | 10           | 2.5          | 6           | 8         |
| Total                      | 38           | 10           | 40          | 59        |

## 2.1 Creating the Worked-Example Videos

For several years prior to (and including) Fall 2022, the CS380 course has incorporated worked examples, primarily in the form of modeled example solutions to in-class exercises (ICEs). Typically, after some amount of lecture, students are given time in class to complete a series of exercises related to the lesson’s content, normally organized as a series of lesson worksheets. The instructor then works out the solution to one or more of the examples live in class. The following lesson, a static copy of the worked-out solutions of the majority of in-class exercises is distributed to students. Distributing solutions in this manner ensured students had access to solutions to in-class exercises that an instructor may not have had time to demonstrate in class. During a typical course execution, students are also told that the quizzes in the course borrow heavily from the concepts covered in the in-class exercises, and that they should primarily focus on the in-class exercise worksheets (and their corresponding provided solutions) as a study aid. The in-class exercises (and their solution files) do not change from year to year.

Armed with this information, the two undergraduate authors created forty worked-example videos of select in-class exercises. All the videos were less than ten minutes in length, with 65% shorter than five minutes. Each worked-example video had a static counterpart, namely the instructor-created worksheet solution PDFs provided by the instructor to all students in CS380. The process to create the worked-example videos was as follows: each student author selected twenty in-class exercises to create videos of, based on their own

Table 2: Population Statistics

| Semester | Population Size ( $N$ ) | "Weak" | "Average" | "Strong" |
|----------|-------------------------|--------|-----------|----------|
| Control  | 44                      | 7      | 21        | 13       |
| Test     | 50                      | 8      | 32        | 7        |

experience of what content was particularly difficult, and from their conversations with their classmates. After selecting the in-class exercises to port, each student author re-did the corresponding in-class exercises, and reviewed the associated static solutions, making sure to understand the problems fully, and consulted the faculty advisor if they had any questions.

Using an iPad to record their voice and their screens, each student slowly walked through each in-class exercise, recording the step-by-step solving process. Once recorded, the students used iMovie, a free editing software, to edit and polish each video and submit it for verification to the faculty advisor. The final stages of editing and publishing the videos involved an iterative verification and editing process, where the advisor gave the student investigators feedback, and the students edited their videos until they were deemed appropriately accurate and detailed. This process was done for all forty worked-example videos, with each video covering a different in-class exercise.

## 2.2 Data Collection

Data was collected over two Fall semester offerings of CS380. The Fall 2022 semester served as the "control" semester, where none of the students surveyed had access to worked example videos. The "test" population was the Fall 2023 semester, where all students had equal access to the worked example videos. In both semesters, students had access to the static worked-example solution PDFs that have always been provided in CS380. Additionally, students in the control and test semesters had access to a series of short instructor-created videos, which mainly contained a summary of the associated readings; however, a small quantity of the videos (especially from the assembly and concurrency unit) included some worked examples of content covered in the textbook.

Table 2 depicts some details about the populations under study; please note that the two student authors who were enrolled in Fall 2022 are excluded from the control population, as they interacted with the worked-example video content, while their classmates did not.

As part of the semester preparation for CS380, the instructor normally looks at the set of incoming students and "flags" certain students based on their performance in the three pre-requisite courses. If a student earned C grades or lower in all their pre-requisite courses, they are flagged as someone

who may struggle in the course. Similarly, if a student has earned A-grades in all the pre-requisite courses, they are flagged as someone who will typically do well. Noticing that the test population had a larger number of struggling students and half the number of strong students as the control population, we partitioned the control and test populations into performance categories for part of our analysis: students in the "strong" partition earned an A- or higher in the pre-requisite courses for CS380; students who struggled and earned a C or lower in their courses were placed in the "weak" partition. Lastly, all other students were placed in the "average" partition. The sum of the three partitions does not add up to the population size; foreign exchange students and non-majors (who did not take the pre-requisites at West Point) were excluded from the partitions, along with one student who repeated the course.

Quantitative and qualitative data were collected for this study. The quantitative data used for this study were primarily quiz grades in the control and test populations. The faculty author has taught CS380 for a number of years and therefore was able to ensure that the difficulty and outcomes tested by each quiz remained consistent between the control and test semesters. Videos were posted on the West Point's Microsoft Stream service; as such, we were also able to use the video view counts tracked by Microsoft Stream to get a rough idea of how often videos were being watched.

Students in CS380 in the test semester were also asked to take an optional, anonymous survey about their experiences with the worked-example videos in CS380, administered at the course midpoint and again at the end. To incentivize responses while maintaining anonymity, CS380 offered two points of extra credit for each survey if at least ninety percent of the students in the course filled out the survey. All surveys were distributed electronically with a consent coversheet that informed students of the risks and benefits of completing the study and included verbiage about the extra-credit incentive. Students had to explicitly consent before being able to view the survey questions.

The survey consisted of seven questions. The first question asked students to rank according to a 5-point Likert scale how helpful they found particular class resources (including the worked-example videos), ranging from "very unhelpful" to "very helpful". The second question asked students to rank (from "most important" to "least important") the study aids they used to succeed on quizzes. The next two questions asked students to self-report the number of worked-example videos they watched and what other not-listed resources students used to study for the course quizzes. The last three questions were open-ended response questions, that asked students what they liked about the worked-example videos, what they felt could be improved about the videos, and how impactful they felt it was to have access to videos created by a peer, vs. an instructor in the course.

### 3 Results

Quiz grades were tabulated over all 10 quizzes in CS380. All available quiz data for the Fall 2022 and Fall 2023 populations were used in this study, with the population statistics summarized in Table 2. Of the 50 students in the test population, 33 students responded to the mid-point anonymous survey (66% response rate), and 39 responded to the survey (78% response rate). Since the surveys were anonymous, it was impossible to associate responses with students in any of the partitions.

#### 3.1 Quiz Performance

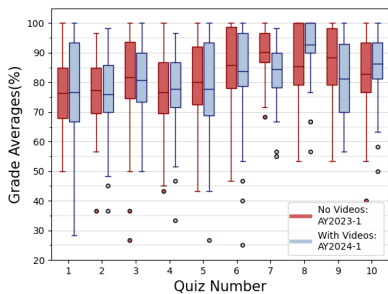
Figure 1 depicts box plots of the grades across all quizzes from the control and test semesters in CS380. Instead of the median, the average quiz grades are indicated by the middle line within each box plot. The box plots (along with the associated interquartile ranges and outliers) were generated using Python's Matplotlib `boxplot()` function.

Figure 1a shows the overall performance of the control and test semesters. As expected, the students in the control semester performed slightly better on the majority of the quizzes on average than those in the test population, owing to the larger number of characteristically weak students and the smaller number of strong students in CS380 in Fall of 2023. However, students in the test semester generally demonstrated slightly better performance on all the quizzes, with higher third quartile performance on three quizzes, higher first quartile performance on five quizzes, and either a higher "max" or "min" score on five quizzes.

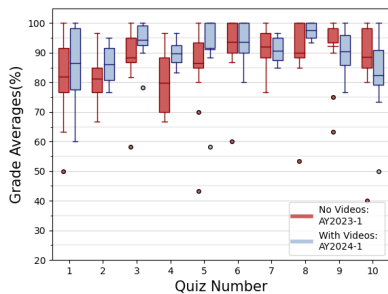
In the strong partition (Figure 1b), students with access to worked-example videos generally performed better than those who did not, having higher averages or third quartiles for six quizzes, and shorter interquartile ranges for five quizzes, suggesting consistently better performance. Interestingly, students in this group did worse than the control group on the last two quizzes, which covered concurrency topics.

In the average partition (Figure 1c), the students with the worked-example videos had first quartiles in six of the ten quizzes. In seven of the ten, the interquartile ranges are shorter in the test population, suggesting that students who had access to worked-example videos generally performed better on the quizzes. We note however that the means for both the control and experimental groups were very similar in the average partition, with differences of less than 4 percent.

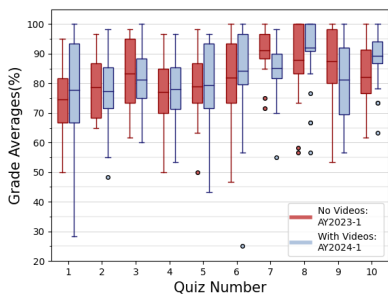
Lastly, in the weak partition (Figure 1d), the students in the test semester performed worse than the students in the control semester in seven of the ten quizzes, regardless of having access to worked solution videos. However, we do



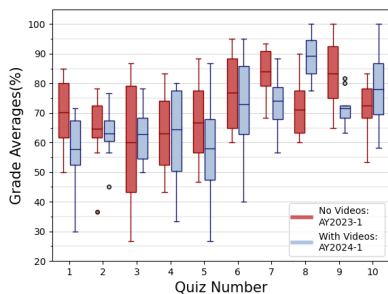
(a) Overall Average



(b) Strong Partition



(c) Average Partition



(d) Weak Partition

Figure 1: Box Plot Comparisons of Quiz Averages across Control and Test Semesters

note that the interquartile ranges were generally smaller in the test semester than in the control semester, suggesting that students with access to worked-example videos in the weak partition had a lower variation in performance.

To better understand our results, we took a closer look at the video view counts and viewers associated with each video. All videos were watched by some fraction of the students; the high numbers of views suggested that students who watched the worked example videos watched them repeatedly, with viewership peaking immediately prior to a quiz. However, the data suggests that high view count of particular videos is primarily a reflection on student uncertainty of the material. Correlating view counts with quizzes suggest that the videos with the highest view counts were associated with quizzes that had some of the lower averages.

Quantitatively, it appears that the worked-example videos had the greatest positive impact on strong students and a modest positive impact on average



and weak students. We note, however, that consistent with prior work, we did not see statistically significant differences in the means of the test and control populations.

### 3.2 Student Perspectives on Worked-Example Videos

To gain student perspectives on the helpfulness of worked-example videos, we asked students in the mid-point and course-end surveys to individually assess the helpfulness of the worked-example videos compared to other named class resources (the static in-class worksheet solutions, instructor-created videos, and the course textbook). We also asked students to rank the perceived usefulness of the aforementioned resources for studying for quizzes. Additionally, we asked them several open-ended questions to get a wide perspective of their answers.

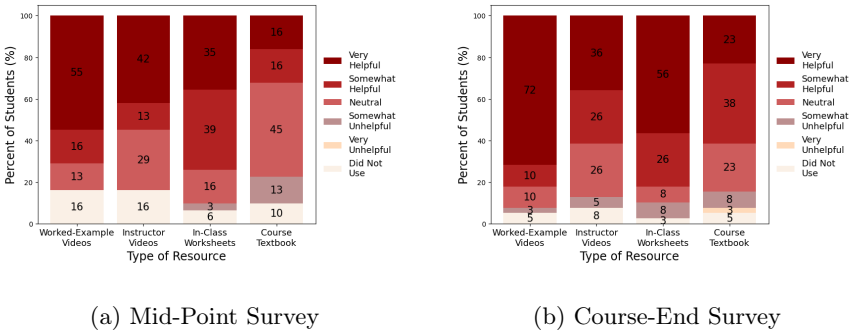


Figure 2: Student Perspective on Helpfulness of Worked-example Videos compared to other class resources

Figure 2 depicts how generally helpful students found worked-example videos and the other available classroom resources at the mid-point and the end of the course. In both the mid-point and course-end survey, the worked-example videos were consistently rated by a majority of students as “very helpful”, with 55% at the mid-point and 72% at the course-end. We believe this is partially due to a lack of awareness of the worked-example videos on the part of some students; 16% of the students indicated that they did not use the worked-example videos in the mid-point survey, while only 5% indicated that they did not use the videos by the end of the semester. We note that the latter half of the course had fewer worked-example videos than the first half; correspondingly, a greater percentage of students reported the in-class worksheet solutions as “very helpful” in the course-end survey compared to the midpoint.

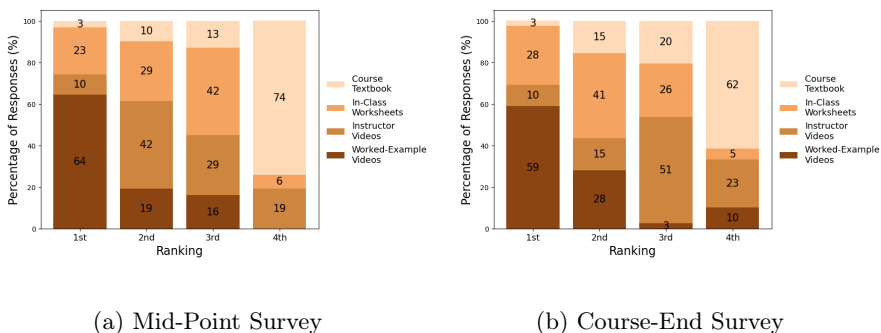


Figure 3: Student Ranking of Usefulness of Worked-example Videos compared to other class resources to succeed on quizzes

Figure 3 shows how students ranked the importance of each aforementioned classroom resource in helping them study and succeed on quizzes. In both surveys, the majority of students rated the worked-example videos as their top study resource, with a smaller fraction rating the static worksheet solutions as their top resource. Only 13% of students pointed to other resources as their top study aid. In short, our results demonstrate an enthusiasm for the video medium amongst the surveyed students.

### 3.3 Open Feedback

The final part of the survey given to the students of the course asked for open feedback on the worked-example videos. Students who reported watching the worked-example videos generally described them as being “thorough” and “well-explained”, and that they appreciated the “step by step explanation” and “pace” of the explanations. *“I like how the videos explain thoroughly how to solve the problems and do not take any shortcuts”,* said one respondent. A few students also appreciated being able to pause and replay components of the worked-example videos: *“I can pause as much as I want and go at my own pace”* said one student.

Students did not offer much constructive feedback for improvement, except for requesting more worked-solutions videos. Some reported challenges accessing the videos due to unfamiliarity with the learning management system, which we plan to work on addressing in the future. A couple of students asked for explanations to be even “slower” and “more nuanced”, highlighting the challenge of generating videos that appeal to all students: *“They are very good how they are”,* said one student, *“but sometimes they skip over a small*

*section or step which may seem trivial, but when learning it is very useful to have each small step."*

Lastly, we asked students how impactful (if at all) having worked-example videos created by peers was rather than those created by instructors. The student respondents were fairly split; 38% felt that peer-created videos had a greater impact, and another 38% felt that it didn't matter. Students who felt that peer-created work-exemplified videos were impactful alluded to how problems and concepts in the videos were explained in a way that were more intuitive to a student who is learning the material for the first time "*I believe the peers understand where other students minds are at and know we do not understand it as well as the instructors do,*" said one student. Students who preferred instructor-made worked-example videos alluded to the thoroughness and organizational qualities of the videos: "*instructor videos are much better planned, thought-out and organized,*" said a student. Another student succinctly stated the feelings of those that felt that the peer aspect did not matter: "*No impact. The fact that there was a video made all the difference.*"

## 4 Lessons Learned & Conclusions

Our results suggest that worked-example videos are an effective study tool that moderately increased the average quiz score of students who used them as a study resource. Additionally, a majority of students rated worked-example videos as a helpful study resource, and also rated the videos as the most important study tool for quizzes overall. In addition to preferring the video modality, several students appreciated the peer-made nature of the videos, and asked for more to be produced. Prior work [11] suggest that the effects of peer-made solutions may also have farther-reaching effects than just performance in the immediate course; one study on the impacts of peer tutoring on tutor and tutee's performance found that the grade point average of tutees increased holistically rather than just in the course they received peer tutoring in [11]. We speculate that worked-example videos can function as a form of peer tutoring which would not only help students understand individual course subjects, but would help key students in on how the tutor thinks about course material. Individuals watching the videos can learn heuristics which translate to other computer science courses and thus increase their overall performance.

Our data also suggests that worked-example videos are an effective study tool for use in intermediate computer science courses such as computer systems courses. Having access to video worked-examples assisted student performance even in non-programming content in CS380, supporting the notion that worked-example videos are useful for a variety of non-programming topics in computing. In the semesters since they became available, the worked-example

videos created by the student authors have become the most popular study resource in the course.

We conclude by offering some perspectives from the student and faculty authors on their experiences of the worked-example video creation process.

#### 4.1 Student Creators: Experiences and Perspectives

While making the videos has continued to help the current student population, the act of *creating* the worked-example videos was incredibly formative for both student authors. We knew that these videos would be shared with peers in our department, and as such, we spent time to ensure that our thoughts were well laid out and concise. It was a time-intensive process, requiring over 80 hours of work to create, edit, and refine our videos.

Video deliverables require particular focus to create, and for the resource to be effective to others, it should be absent of erroneous content. Because this process was so lengthy and we had to understand the material at such a high level, the task of answering static questions on in-class quizzes, without the added pressure of narration and editing, was substantially easier. Quantitatively, we both ended the course with the highest letter grade, an A+, but more importantly, the process ensured we were incredibly confident in the material when we were quizzed.

Additionally, this study developed our understanding of basic video editing software, a skill that is translatable to other components of traditional college education such as group presentations and final projects. While not initially apparent, learning basic video editing skills has been helpful in a variety of different academic environments [2]. Furthermore, learning to teach and present material is a critical skill that not only improves an individual's understanding of the material but also develops critical presentation and interpersonal skills that are translatable across multiple disciplines.

#### 4.2 Instructor Experience and Perspective

Based on prior work, the faculty author had two predictions: first, that having students create worked-example videos would be beneficial to their individual learning, and second, by offloading the work of video creation onto students, faculty time will be freed up to do other tasks.

In retrospect, having only two students (rather than the entire course) participate in the worked-example video creation process was important. As our surveys have shown, creating concise and well-explained worked-example videos is challenging, and requires some amount of effort. The faculty author spent quite a lot of time with the two students ensuring that video content was free of incorrect explanations and assumptions. That iterative process,

while extremely valuable for the two student authors, was exceptionally time-consuming for the faculty author. In retrospect, it would have taken the faculty author less time to generate the worked-example videos on their own. However, the benefits to the students creating the videos (and to their peers who watched them) is compelling, and would have been undoubtedly lost.

There is an open question on the value of "crowd-sourcing" worked-example videos from the general student body during a particular course iteration. The perceived benefit would be that more students would have the opportunity to gain the insights offered through the video-creation process, like the students authors. On the other hand, creating good quality worked-example videos is time-consuming and difficult. The two student authors cared deeply about helping their peers and doing a good job; this is not always true of the average student. The more students involved in video production in a semester, the more faculty oversight that is required to make sure information is accurate. From the faculty author's perspective, the greatest challenge in producing high-quality peer-created videos is finding students who are willing and capable of producing high-quality study resources. West Point does not have undergraduate teaching assistants; the student authors however were popular tutors in the department.

Lastly, course designs involving well-established in-class problem sets like CS380 likely benefit the most from the worked-example video creation process, as it is a one-time operation whose products can be used over future semesters. For courses under active revision, creating high-quality worked-example videos that may ultimately be discarded might be judged as a poor use of time. However, our results show that the video modality for worked examples involving non-programming content is perceived as being very valuable to students, and that peer-created worked-example videos are rated as extremely valuable by both the peer creators and the student consumers alike.

## Acknowledgement

This project is supported by the National Science Foundation (DUE - 2141814). The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Army, Department of Defense or the U.S. Government.

## References

- [1] Robert K Atkinson, Alexander Renkl, and Mary Margaret Merrill. "Transitioning from studying examples to solving problems: Effects of self-

- explanation prompts and fading worked-out steps.” In: *Journal of educational psychology* 95.4 (2003), p. 774.
- [2] Christa Chewar and Suzanne J. Matthews. “Lights, camera, action! video deliverables for programming projects”. In: *Journal of Computing Sciences in Colleges* 31.3 (2016), pp. 8–17. URL: <https://dl.acm.org/doi/10.5555/2835377.2835380>.
  - [3] Michelene T.H. Chi et al. “Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems”. In: *Cognitive Science* 13.2 (1989), pp. 145–182. URL: [https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1302\\_1](https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1302_1).
  - [4] Derek Hwang et al. “A Qualitative Analysis of Lecture Videos and Student Feedback on Static Code Examples and Live Coding: A Case Study”. In: *Proceedings of the 23rd Australasian Computing Education Conference*. ACE ’21. Virtual, SA, Australia: Association for Computing Machinery, 2021, pp. 147–157. ISBN: 9781450389761. URL: <https://doi.org/10.1145/3441636.3442317>.
  - [5] Suna Kyun, Slava Kalyuga, and John Sweller. “The Effect of Worked Examples When Learning to Write Essays in English Literature”. In: *The Journal of Experimental Education* 81.3 (2013), pp. 385–408. URL: <https://doi.org/10.1080/00220973.2012.727884>.
  - [6] Bridjet Lee and Kasia Muldner. “Instructional Video Design: Investigating the Impact of Monologue- and Dialogue-Style Presentations”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI ’20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–12. ISBN: 9781450367080. URL: <https://doi.org/10.1145/3313831.3376845>.
  - [7] Bruce McLaren and et al. “The Efficiency of Worked Examples Compared to Erroneous Examples, Tutored Problem Solving, and Problem Solving in Computer-Based Learning Environments”. In: *Computers in Human Behavior* 55 (Feb. 2016), pp. 87–99. URL: <https://www.sciencedirect.com/science/article/pii/S074756321530114X>.
  - [8] Briana B. Morrison. “Dual Modality Code Explanations for Novices: Unexpected Results”. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER ’17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 226–235. ISBN: 9781450349680. URL: <https://doi.org/10.1145/3105726.3106191>.

- [9] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. “A Review of Worked Examples in Programming Activities”. In: *ACM Trans. Comput. Educ.* 23.1 (Dec. 2022). DOI: 10.1145/3560266. URL: <https://doi.org/10.1145/3560266>.
- [10] Alexander Renkl and Robert K Atkinson. “Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective”. In: *Cognitive Load Theory*. Routledge, 2016, pp. 15–22.
- [11] Eun Hee Seo and Min Ji Kim. “The Effect of Peer Tutoring for College Students: Who Benefits More from Peer Tutoring, Tutors or Tutees?” In: *The New Educational Review* 58 (2019), pp. 97–106. DOI: <https://doi.org/10.15804/tner.19.58.4.07>.
- [12] Ben Skudder and Andrew Luxton-Reilly. “Worked Examples in Computer Science”. In: *Sixteenth Australasian Computing Education Conference (ACE 2014)*. Vol. 148. ACS. Auckland, New Zealand: Australian Computer Society (ACS), Jan. 2014, pp. 59–64. URL: <https://dl.acm.org/doi/10.5555/2667490.2667497>.
- [13] Ben Stephenson. “Coding Demonstration Videos for CS1”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 105–111. ISBN: 9781450358903. URL: <https://doi.org/10.1145/3287324.3287445>.
- [14] John Sweller and Graham A. Cooper. “The use of worked examples as a substitute for problem solving in learning algebra”. In: *Cognition and Instruction* 2.1 (1985), pp. 59–89. DOI: 10.1207/s1532690xci0201\_3. URL: <https://www.jstor.org/stable/3233555?seq=2>.
- [15] Albina Zavgorodniaia et al. “Algorithm Visualization and the Elusive Modality Effect”. In: *Proceedings of the 17th ACM Conference on International Computing Education Research*. ICER 2021. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 368–378. ISBN: 9781450383264. URL: <https://doi.org/10.1145/3446871.3469747>.
- [16] Rui Zhi et al. “Exploring the Impact of Worked Examples in a Novice Programming Environment”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 98–104. ISBN: 9781450358903. URL: <https://doi.org/10.1145/3287324.3287385>.

# Enhancing Empathy and Inclusivity in Computer Science Education: An Empirical Study on Accessibility Interventions for Undergraduate Students\*

Jami Cotler<sup>1</sup>, Eszter Kiss<sup>2</sup>, Dmitry Burshteyn<sup>1</sup>,  
Megan Hale<sup>3</sup>, Amani Walker<sup>1</sup> and John Slyer<sup>4</sup>

<sup>1</sup>Siena College

{jcotler, dburshteyn, a27walk}@siena.edu

<sup>2</sup>James Cook University

Brisbane, Australia

eszter.kiss@jcu.edu.au

<sup>3</sup> Brockport College

megaloo52911@gmail.com

<sup>4</sup> SkyHigh Adventures

SKYHIGHadventures@gmail.com

## Abstract

This paper introduces an innovative approach in computer science education that centers on building empathy and inclusivity skills, particularly towards designing accessible technology for individuals with disabilities. In a landscape where 41 states in the United States (82% of the US) have embraced accessibility policies, this study introduces in-person accessibility interventions for undergraduate computing students, focusing on Human-Empathy Accessibility Learning (HEAL). What makes this approach innovative is its emphasis on abilities-based experiential learning and direct engagement with inclusivity challenges. The interventions

---

\*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.



are structured to cultivate empathy as a foundational skill for effective communication, collaborative problem-solving, and inclusive design in technology. This study evaluates the effects of these novel interventions on students' empathy, career goals, and attitudes towards accessibility through three experimental groups from diverse academic settings, including a control group. Key to our methodology is the implementation of an "empathy lab", where students actively engage in empathy-driven experiences, including a unique collaboration with a woman who is blind for one of the groups. We employ a mixed-methods research design, leveraging the Perth Empathy Scale (PES) to quantify empathy development and analyze student reflections to understand shifts in attitudes, skills, and design philosophy. The findings reveal a marked improvement in empathy, skill acquisition, and a renewed commitment among students to incorporate accessibility into their future technology endeavors. This study is a testament to the transformative potential of integrating empathy and accessibility into computer science curriculum, paving the way for a more inclusive and ethically mindful technological landscape.

## 1 Introduction

In an increasingly interconnected and diverse world, the call for greater inclusivity and accessibility has become more pronounced than ever before. Within this evolving societal landscape, it is important that undergraduate college students not only acquire academic knowledge but also cultivate a deep understanding of the diverse perspectives and needs of their fellow citizens. In Zhou's [19] critical review on "Empathy in Education" emphasizes the importance of empathy in the educational context, suggesting its pivotal role in shaping students' understanding of diverse perspectives and needs, particularly in technology-related fields. This perspective forms the backdrop of our research, which focuses on the abilities individuals with disabilities have and the challenges they face in technology interactions—a crucial issue often not covered in conventional computer science curricula [18]. A construct such as empathy is ultimately defined by the way it is measured. In this study, using the PERTH Empathy Scale empathy is defined as cognitive and affective empathy. Cognitive empathy allows for better communication and problem-solving, particularly in social and professional contexts, while affective empathy fosters deeper emotional connections and can drive altruistic behavior [4, 11]. To address this crucial aspect of holistic education, this study investigates the impact of accessibility workshops on empathy, with a particular focus on ability-based design [18], evaluating skills development and attitudinal change among undergraduate college students in the context of their career development. We consciously shift from traditional disability simulations, criticized for perpetuating stereotypes [15], to interactive methods co-developed with an elite triathlete

who is blind to offer a more authentic experience. With this pilot study we aim to shed light on the transformational potential of ability-based empathy workshops, which have emerged as a promising tool in fostering greater inclusivity and understanding among young adults. As the undergraduate years serve as a pivotal period for personal and professional development, our study seeks to explore how accessibility interventions can equip students with the skills and perspectives necessary to navigate an increasingly diverse workforce and contribute positively to society. This research endeavors to answer several key research questions. Do the Human-Empathy Accessibility Learning (HEAL) interventions lead to increased empathy and attitudinal changes among college students, encouraging them to adopt more inclusive mindsets? Do these changes impact career development trajectories? By addressing these questions, our study aims to provide valuable insights into the potential benefits of empathy-driven education, ultimately contributing to the cultivation of more compassionate, inclusive, and empathetic technical professionals.

## 2 Literature Review

In the ever-changing field of computer science (CS) education, there is a growing consensus on the critical need to incorporate empathy and accessibility into the curriculum. This need is underscored by the significant educational gap in CS, where emphasis on understanding and designing for users with disabilities is often lacking [2, 3, 18]. Empathy in CS education, encompassing both cognitive and affective components, is essential for understanding the varied needs of a diverse user base, including those with disabilities [5, 7]. However, traditional methods of cultivating empathy, such as disability simulations, have been scrutinized for potentially reinforcing stereotypes and failing to provide an authentic understanding of the disability experience [15]. The discourse in current literature suggests that alternative educational interventions are necessary to effectively instill empathy in CS students. For instance, immersive virtual reality experiences have been shown to empower students by providing a more profound insight into the challenges faced by individuals with disabilities [1]. Similarly, experiential learning approaches have been identified as beneficial in providing students with firsthand experience in accessibility challenges and solutions [10]. These methods offer a deeper and more empathetic understanding of accessibility issues, promoting more inclusive design practices [9, 17]. Additionally, the incorporation of Universal Design principles in educational settings can create a more inclusive learning environment, benefiting all students and highlighting the importance of accessible technology design [6]. The concept of empathy-centric design in industry, where empathy is integrated into professional practices, further emphasizes the importance of

understanding user needs in technology development [8]. The literature indicates that while the importance of teaching empathy and accessibility in CS education is evident, there is a critical need to move beyond traditional disability simulation methods. More authentic and interactive approaches, such as immersive technologies and experiential learning, can better equip students to create technology that is both accessible and inclusive.

### 3 Methodology

This research evaluates the effects of two accessibility-focused interventions on undergraduate students' empathy, career aspirations, and attitudes towards accessibility. Participants comprised three groups: two from a small liberal arts college and one from a top-tier technical university. The first group acted as a control, receiving no intervention. The other two groups participated in an immersive "empathy lab" experience that was co-developed with a woman who is blind, with one group collaborating with the same woman in a client scenario. In the immersive "empathy lab," students engaged with five stations featuring assistive technologies like screen readers and alt tag exercises. The "reveal" intervention involved students initially creating a basic website for an elite athlete. Upon meeting her and learning about her abilities as someone who is blind, they were then challenged to redesign the website. This pilot study employed a mixed-methods approach, incorporating both quantitative and qualitative analyses, to offer a comprehensive understanding of the interventions and the impact on participants across different settings. The decision to use mixed methods was strategic, aiming to thoroughly investigate the experimental findings while navigating the challenges and complexities of conducting research in educational settings. Obtaining ethics approvals for studies in these contexts may often be a time-consuming and difficult process; thus, it was important to address a broad range of research questions without sacrificing methodological standards or compromising experimental controls. Specifically, we intended to compare the effects of combined interventions against single interventions and a control group. Additionally, we examined how varying college populations influenced training outcomes. This allowed us to better understand how each intervention affected participants within specific contexts. Empathy growth was measured using the Perth Empathy Scale or PES [5]. This instrument was administered to each student in the study at the beginning of the semester and again at the end of the semester. This self-report scale consisted of 20 items designed for assessing cognitive and affective empathy, including their valence-specific aspects. It comprised of items that gauge the accurate recognition of others' emotions (cognitive empathy) and the emotional resonance with others (affective empathy), covering both negative and positive emotions,

resulting in four theoretical subscales, each containing five items without the need for reverse-scoring, in alignment with recommended practices [16, 14]. Both cognitive and affective empathy are crucial for effective interpersonal interactions. A six questions reflection was distributed at both institutions after gaining ethics approval at both research sites. The reflections asked about Q1: Attitude and Perspective Change, Q2: Skills Acquired, Q3: Immediate Changes Planned, Q4: Future Approach to Design, Q5: Professional Applications, and Q6: Workshop Feedback. The reflections were administered after each intervention.

### 3.1 Participants

The study involved 69 undergraduate college students. 64 participants provided a complete set of responses used to conduct quantitative analysis. At a tier 1 technical university (referred to as institution A), an empathy lab (intervention 1) was conducted with 31 participants. At a small liberal arts college (institution B), an empathy lab (intervention 1), along with the “client reveal” (intervention 2) was conducted with 15 participants. Another group of 24 participants at institution B served as the control group. Ethics approval was granted for both institutions and participation in the study was an optional part of each class. The distribution of the sample is summarized in Table 1. The number of students in each class was determined by students self-selecting a particular class.

| Interventions                | Institution A – Empathy Lab | Institution B – Empathy Lab | Institution B – Client Reveal | Institution B – Control |
|------------------------------|-----------------------------|-----------------------------|-------------------------------|-------------------------|
| Total number of participants | 31                          | 14                          | 15                            | 25                      |
| Gender - female              | 39%                         | 21%                         | 20%                           | 12%                     |
| Gender - male                | 61%                         | 79%                         | 80%                           | 88%                     |

Table 1: Sample Distribution

## 4 Findings

In this mixed methods study we sought to compare the effects of combined interventions against single intervention and a control group. Additionally, we examined how varying college populations influenced training outcomes. This allowed us to better understand how each intervention affected participants within specific contexts. In educational settings, where strict experimental control is challenging to maintain, qualitative measures proved to be invaluable for

capturing the nuances and complexities associated with the applied interventions. We hypothesized that Human-Empathy Accessibility Learning (HEAL) interventions lead to increased empathy and attitudinal changes among college students and impact career development. Our quantitative results indicate marginal significance. The dual-intervention condition showed the most benefit, followed by the single-intervention condition. In contrast, no significant changes were observed in the control group. The mean overall differences in pre-to-post empathy scores were pointing towards potential effects, although our Analysis of Variance (ANOVA) did not yield statistically significant differences between the experimental and control groups in Total Empathy Score. However, we observed marginal significance in individual items, suggesting an increase in self-awareness and skills in recognizing emotions in others.

- "When I see or hear someone calm, it makes me feel calm too."  $F(1,63) = 3.72, P < .058$
- "When I see or hear someone angry, it makes me feel angry too."  $F(1,63) = 3.346, P < .072$
- "Just by seeing or hearing someone, I know if they are feeling calm."  $F(1,63) = 2.825, P < .098$

ANOVA is widely used in educational research, especially in experiments involving multiple groups and has been shown to be robust even with smaller sample sizes. Our observation of marginal significance despite small sample sizes, resulted in ANOVA detecting overall trends [13], although it requires careful interpretation of results [12]. Our quantitative results were substantiated by our qualitative findings, which enriched the trends observed in the numerical data. While students benefited in both settings, the advantages supported by our qualitative data appeared to be unique to each educational context.

#### 4.1 Findings at Institution A (Empathy Lab)

In the study, 71% of participants reported changed views about people with impairments after an empathy lab workshop, with 32% noting a large change and 23% a slight one. Twenty-six percent saw no change, and 3% were neutral. Fifty-seven percent developed empathy, stating they could “see the world through their eyes” and felt “much more sensitive to them and the daily problems they face.” One participant remarked, “I used to think that designing for accessibility was just a box to check off... But after this workshop, I realized it’s also really smart business.” Twenty-nine percent reflected on accessibility

thinking. Eighty-three percent gained new skills, while 16% improved awareness but did not acquire new skills. A participant said, "The workshop really led me to experience what some people are going through... I think this will help better design websites that cover all groups of people." All respondents agreed to update their team's website design for better accessibility, with 16% focusing on fonts and inclusive features. Ninety-seven percent planned to incorporate accessibility in future technology designs, with 39% prioritizing user needs and inclusivity. Ninety-three percent had clear plans to apply these learnings professionally, focusing on empathy (19%) and inclusion (64%). A participant shared, "I can apply my better developed skills of empathy... and remind my team to design with everyone in mind." Nineteen percent planned to advocate for accessibility in the workplace. Another participant noted the importance of learning new languages for effective communication. The students appreciated the chance to "see the challenges some people face" and learn about screen readers. Summarizing the experience, one said, "I obtained new skills, broadened my awareness of accessibility and inclusivity, and established a plan of action for applying these concepts into my professional practice."

## **4.2 Findings at Institution B**

Secondly, the reflections gathered at the small liberal arts college were collated here in two parts: findings on the first intervention (Empathy Lab workshop) and findings on the second intervention (immersive workshop with the elite athlete).

### **4.2.1 Empathy Lab**

In response to the question about changing attitudes towards people with impairments, 93% of 15 participants reported a change in perspective towards people with impairments after an intervention, with 7% neutral. Most (64%) cited increased empathy, saying "Walking in their shoes...has deepened my empathy for them." They acknowledged the importance of accessible design, as one noted, "Engaging in these exercises... made me realize how vital it is to make systems easier and more accessible." Regarding skills acquired, 71% mentioned technical abilities like improved image descriptions for accessibility, and half developed soft skills, particularly empathy. All participants agreed to make their websites more accessible, including revising alt tags and improving color contrast. Future technology design will focus on accessibility and usability, with 43% planning people-specific changes and 64% tech-specific adjustments. Participants aim to advocate for accessibility in professional settings and create technology enjoyable for all. Ninety-three percent had plans to implement these insights professionally, emphasizing usability testing and

accessibility in future jobs. Some pledged to advocate for accessibility in their workplaces. Feedback on the workshop was positive, with 78% finding it educational in raising disability awareness and accessibility needs. Some wished for more in-depth activities and resources. The workshop was valued for its practical application in future design projects.

#### 4.2.2 Client Reveal

When asked if their views on people with impairments changed after learning about the client's condition, 71% of participants said yes, 7% saw no change, and 21% were neutral. Those whose views changed noted increased respect for the challenges faced by people with impairments in using technology, like screen readers. One respondent said, "something you work with daily" could be frustrating for visually impaired people. Another expressed empathy, saying, "spending some time in their shoes for a couple of minutes was really frustrating." All fourteen participants reported skill development during the workshop, including using screen readers (71%) and keyboards (7%). Twenty-two percent felt their empathy skills improved, with one saying, "Getting to work with it and understanding the difficulty helps me see things from the point of view of others who may need this technology." Participants planned to apply these skills in the future, with one noting, "I'll be more conscious of these things and try exercises to put myself in the place of other people." They all agreed to update their website for better accessibility, with plans to use better headers, efficient screen readers, and descriptive alt tags. One participant remarked, "I learned how important it is to incorporate accessibility into our websites." Regarding future technology design, 71% aimed for greater accessibility, while 28% said changes would depend on projects and company expectations. Fifty-eight percent planned to accommodate diverse needs. One said, "I will improve on what I know and create all my future technologies to be accessible." Ninety-three percent knew how to apply workshop learnings professionally, with 28% considering employer expectations and 21% focusing on informing others about accessibility. One shared, "I plan on being a greater advocate... just being aware of the many different needs that your users/customers may have is very important for maintaining user satisfaction." Feedback on the workshop was positive, with 93% finding it useful. Participants appreciated demonstrations like using screen readers and navigating websites. They expressed a desire to learn more about screen readers, including their use in desktop applications.

## 5 Discussion

This section discusses qualitative data analysis, frequency analysis, overall summaries and comparisons.

### 5.1 Quantitative Data Analysis

Our quantitative findings support the trend hypothesized in this study. Our analysis yielded a more significant improvement in empathy for the dual-intervention group than the single-intervention group. The mean differences showed the larger pre to post changes in empathy in the dual-intervention group followed by our one-intervention group with no differences in the control group. However, Analysis of Variance yielded no statistical significance between groups for the overall empathy measure while showing marginal significance for several specific items. Including different college populations for comparison allowed us to explore the generalizability of our findings and the impact of contextual variables.

### 5.2 Frequency Analysis

The impact of the "empathy lab" intervention yielded several findings. Table 2 presents the results of a frequency analysis based on the reflection questions related to the impact of the accessibility interventions. Overall, the table indicates that the interventions had an overall positive impact on participants, leading to change in their views, the development of skills, and a strong willingness to make changes and incorporate accessibility and inclusivity into their future work and professional lives.

| Questions  | Frequencies   |     |           |  |
|--|---|-----|-----------|--|
| Q1: Changing views and developed empathy         | 71% change in their views   | 57% | claimed   | developed empathy                            |
| Q2: Skills developed                             | 32% learnt using screen readers                                   | 10% | improved  | communication skills                         |
| Q3: Willingness to make change                   | 100% agreed to implement changes                                  |     |           |  |
| Q4-5: Future approach & Professional application | 97% positive about incorporating accessibility in future projects | 64% | wanted to | integrate inclusivity into professional life |
| Q6: Workshop effectiveness                       | 87% found it effective  | 35% | desired   | more time per station                        |

Table 2: Frequency Analysis



### 5.3 Qualitative Data Analysis

Qualitative data were analyzed for all interventions. A Kappa score of 0.69 indicated substantial inter-rater agreement. We recommend merging the following two nodes "accessibility product nodes" and "assisted technology dimension" in future studies for higher reliability.

### 5.4 Summary of Qualitative Results from Institution B (Empathy Lab)

After the intervention, 93% of participants reported a change in their perspective towards individuals with impairments. A significant number (64%) highlighted empathy as the primary driving force behind this change. Participants universally agreed that understanding the challenges of impaired individuals is crucial, both personally and professionally. Most participants (71%) gained specific technical skills, including improved image descriptions, effective use of alt tags, and color selection aimed at accessibility. Soft skills like teamwork and empathy were also notably developed, as mentioned by 50% of the participants. All participants expressed the intention to make practical changes to their teamwork websites. These changes include revising alt tags, enhancing color contrast, and making content more concise. All respondents planned to change their future design approach. While 43% of the responses focused on people-specific changes, 64% touched on tech-specific ideas. Advocacy, awareness, and education were mentioned as important tools for planning future accessible designs. Technical factors like navigation and alt tag effectiveness were also highlighted. A substantial 93% of participants planned to incorporate lessons learned about accessibility and inclusion in their professional life. This ranged from specific design considerations to broader approaches like advocacy for inclusion and diversity within their workplaces. Most participants (78%) found the empathy lab useful. A portion (43%) wished for more time to engage in workshop activities. The sessions were lauded for their educational value and the deep insight they provided into accessibility needs. Table 3 indicates that the Empathy accessibility intervention were successful in fostering attitude change, skill development, and a strong commitment to implementing accessibility in both professional and academic contexts.

In summary, the institution B intervention was largely successful in shifting attitudes, imparting new skills, and inspiring future action for more inclusive and accessible design. The empathy lab workshop received positive feedback for its role in this transformation, although some participants felt that more time could have been allotted to each activity.

| Topics                           | Findings   |  |
|----------------------------------|--|--|
| Attitude and perspective change  | 93% change in their views  | 64% claimed developed empathy  |
| Skill development                | Hard skills: image descriptions, alt tags, color selection         | Soft skills (50%)  |
| Intention to implement changes   | 100% agreed to implement changes                                   | Features to look at: alt tags, color contrast, concise content                               |
| Future approaches                | 100% positive about incorporating accessibility in future projects | Features to look at: advocacy, awareness, and education (43%), navigation and alt tags (64%) |
| Application in professional life | Accessibility (93%)  | Specific design considerations, <u>advocacy</u> and workplace diversity                      |
| Workshop effectiveness           | 78% found it useful  | Wished for more time (43%)   |

Table 3: Summary of qualitative data analysis at institution B

## 5.5 Summary of Qualitative Results from Institution A

71% of participants reported a change in their views on people with impairments. About one-third of those participants felt a substantial shift, and 57% claimed to have developed empathy. The workshop also spurred thoughts on equity vs. equality, smart business, and the technical aspects of accessibility. On the hard skills front, 6% learned about accessibility features, 13% about assistive technology and alt tags, 32% about using screen readers, and another 6% about color contrast and accessibility testing tools. Regarding soft skills, 6% noted improvement in teamwork and 10% in communication. All participants planned to implement changes, from enhancing general accessibility (16%) and user interface tweaks like font adjustments (16%) and layout (19%), to improved navigation and content. A strong focus was placed on alt tags (35%) and inclusivity (13%). Almost all participants (97%) were positive about making accessibility a standard practice in future projects. They highlighted various aspects, from involving people with impairments in testing (22%) to focusing on specific tools like alt tags (22%) and color contrast (19%). The biggest takeaways were around empathy (19%) and inclusion (64%). A notable 19% planned to be advocates for accessibility in their professional settings. Participants also saw the direct benefits of accessibility, not just for ethical reasons but also for business expansion. A large majority (87%) found the workshop effective, informative, and engaging. However, 35% wished for more time at each station, and 29% desired more real-world examples.

## 5.6 Comparison of Qualitative Data at Institution A and B (Empathy Lab)

Table 3 provides a comparison of qualitative data between two Institutions (A & B) regarding the impact of accessibility interventions. While both institutions reported positive outcomes from the accessibility workshops, there were differences in the nature of the changes observed. Institution A had a more

| Topics                           | Findings  |   |
|----------------------------------|---|---|
| Attitude and perspective change  | 71% change in their views   | 57% claimed developed empathy   |
| Skill development                | Hard skills: accessibility features (6%), assistive technology and alt tags (13%), screen readers (32%), color contrast, testing tools (6%).<br>Soft skills: communication (10%), teamwork (6%) |   |
| Intention to implement changes   | 100% agreed to implement changes  | Features to look at: general accessibility (16%), fonts (16%), layout (19%), alt tags and inclusivity (13%) |
| Future approaches                | 97% positive about incorporating accessibility in future projects   | Features to look at: testing (22%), alt tags (22%), color contrast (19%)                                    |
| Application in professional life | Empathy (19%) and inclusion (64%)   | Advocacy (19%)  |
| Workshop effectiveness           | 87% found it effective  | Wished for more time (35%), more real-world examples (29%)  |

Table 4: Summary of qualitative data analysis at institution A

diverse attitude change and focused on technology specific aspects, while institution B primarily experienced changes centered around empathy and had a strong specific technical improvement. Both institutions expressed a high intention to implement changes and positive outlooks for future approaches and professional application of the intervention content, albeit with varying emphases.

| Questions                        | Institution A  | Institution B   |
|----------------------------------|--|---|
| Attitude and perspective change  | 71% reported a change with more diverse focus                | 93% reported change mostly centered around empathy                          |
| Skill development                | 16% reported on gaining soft skills                          | 71% reported on gaining technical skills                                    |
| Intention to implement changes   | 100% agreed with more distributed focus                      | 100% agreed with strong emphasis on revising alt tags and color contrast    |
| Future approaches                | 97% positive about changes focusing on tech-specific aspects | 100% positive about changes with focus on tech- and people-specific aspects |
| Application in professional life | 69% application for inclusion                                | 93% concrete plans for implementing changes                                 |
| Workshop effectiveness           | 87% effective workshop                                       | 78% useful workshop   |

Table 5: Comparison of qualitative data

In summary, both workshops were effective in changing attitudes and imparting skills related to accessibility and inclusion, but they differed in focus. Institution B had a more emphatic and holistic impact on participants, while Institution A offered a more technical, diverse set of learnings. Both had almost unanimous agreement on the need for practical changes in design and future approaches.

## 6 Discussion

In conclusion, despite the challenges of a small sample size, this study effectively demonstrated the impact of ability-based empathy and accessibility interventions in different educational settings. The study's strength lies in its mixed-methods approach, integrating both quantitative and qualitative analyses to provide a comprehensive understanding of the interventions' effects. Conducting the study across diverse academic institutions (a Tier 1 technical university and a small liberal arts college) adds significant value. It allowed for the comparison of interventions across different educational cultures and student populations, enhancing the generalizability and relevance of the findings. The implementation of unique ability-based interventions, such as the empathy lab and the "client reveal," offers fresh insights into empathy and accessibility interventions in educational settings. The comparative analysis of these interventions contributes to the growing body of knowledge on effective empathy and accessibility skill development techniques. The combination of quantitative and qualitative methodologies strengthens the study. While the quantitative data provides a solid statistical basis for evaluating the interventions, the qualitative data adds depth and context, capturing nuanced aspects of student experiences that numbers alone cannot convey. Despite the absence of statistically significant differences in total empathy scores, the marginal significance observed in specific items is noteworthy. These findings suggest potential effects on students' self-awareness and emotional recognition skills. Such trends, although not conclusive, indicate the positive impact of the interventions and warrant further investigation. The qualitative findings corroborate and enrich the quantitative trends. They provide valuable insights into how students in different settings uniquely benefited from the interventions. This aspect of the study highlights the importance of qualitative measures in educational research, especially in understanding complex, context-dependent phenomena. The observed trends, coupled with the qualitative feedback, suggest that with a larger sample size, more definitive conclusions might be drawn. This indicates the potential for a significant impact of such interventions on empathy and accessibility development in educational settings. In summary, this study provides valuable insights into empathy and accessibility interventions in educational settings. It lays the groundwork for future research and underscores the potential of such interventions in enhancing emotional skills, attitudinal changes towards accessibility, and inclusivity in career development in diverse environments.

## 7 Limitations and Future Research

The limitations of this study are characteristic of those that often arise in studies utilizing a convenience sample. These limitations include a small sample size, an unequal distribution of students across classes, and the inability to randomly assign participants to different conditions. Additionally, the inclusion of a second campus increases the generalizability of the findings, but it also introduces a threat to internal validity. Specifically, the new institutional environment may add uncontrolled variables that could impact the results, particularly given the absence of random assignment of participants.

### 7.1 Future Research

To increase validity of our findings, future studies should consider increasing the sample size. This would enhance the statistical power of the analyses and reduce the impact of potential confounding variables. Additionally, integrating a control group within each educational institution could offer better insights into the effectiveness of the interventions in specific settings. Given the promising preliminary results, a longitudinal study examining the long-term efficacy of these interventions is recommended. Such an approach would enable us to track the impact of the training not only throughout the college experience but also into the participants' professional lives post-graduation.

## References

- [1] Rhodora Abadia, James Calvert, and Ratna Dasika. “Effectiveness of using an immersive and interactive virtual reality learning environment to empower students in strengthening empathy and mastery learning”. In: *27th International Conference on Computers in Education, ICCE 2019*. Asia-Pacific Society for Computers in Education. 2019, pp. 495–504.
- [2] Catherine M Baker, Yasmine N El-Glaly, and Kristen Shinohara. “A systematic analysis of accessibility in computing education research”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 107–113.
- [3] Catherine M Baker et al. “Including accessibility in computer science education”. In: *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*. 2022, pp. 1–5.
- [4] Simon Baron-Cohen and Sally Wheelwright. “The empathy quotient: an investigation of adults with Asperger syndrome or high functioning autism, and normal sex differences”. In: *Journal of autism and developmental disorders* 34 (2004), pp. 163–175.
- [5] Jack D Brett et al. “The psychometric assessment of empathy: Development and validation of the Perth empathy scale”. In: *Assessment* 30.4 (2023), pp. 1140–1156.

- [6] Sheryl Burgstahler. “Universal Design in Education: Principles and Applications.” In: *DO-IT* (2009).
- [7] Benjamin MP Cuff et al. “Empathy: A review of the concept”. In: *Emotion review* 8.2 (2016), pp. 144–153.
- [8] Luce Drouet, Froukje Sleswijk Visser, and Carine Lallemand. “Using Empathy-Centric Design in Industry: Reflections from the UX Researcher, the Client, and the Method Expert”. In: *Proceedings of the 2nd Empathy-Centric Design Workshop*. 2023, pp. 1–9.
- [9] Mexhid Ferati and Bahtijar Vogel. “Accessibility in web development courses: A case study”. In: *Informatics*. Vol. 7. 1. MDPI. 2020, p. 8.
- [10] Yasmine El-Glaly et al. “Presenting and evaluating the impact of experiential learning in computing accessibility education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. 2020, pp. 49–60.
- [11] Martin L Hoffman. “Empathy and moral development”. In: *The annual report of educational psychology in Japan* 35 (1996), pp. 157–162.
- [12] Timothy R Levine and Craig R Hullett. “Eta squared, partial eta squared, and misreporting of effect size in communication research”. In: *Human Communication Research* 28.4 (2002), pp. 612–625.
- [13] Tamanna Motahar et al. “Building “design empathy” for people with disabilities: An unsolved challenge in HCI education”. In: *Proceedings of the 5th Annual Symposium on HCI Education*. 2023, pp. 68–71.
- [14] Brett A Murphy et al. “Strengths and weaknesses of two empathy measures: A comparison of the measurement precision, construct validity, and incremental validity of two multidimensional indices”. In: *Assessment* 27.2 (2020), pp. 246–260.
- [15] Michelle R Nario-Redmond, Dobromir Gospodinov, and Angela Cobb. “Crip for a day: The unintended negative consequences of disability simulations.” In: *Rehabilitation psychology* 62.3 (2017), p. 324.
- [16] Eric van Sonderen, Robbert Sanderma, and James C Coyne. “Ineffectiveness of reverse wording of questionnaire items: Let’s learn from cows in the rain”. In: *PloS one* 8.7 (2013), e68967.
- [17] Maria Jose Vilches Gonzalez et al. “Developing Empathy towards Experiences of Invisible Disabilities Through Games”. In: *Proceedings of the 2nd Empathy-Centric Design Workshop*. 2023, pp. 1–8.
- [18] Jacob O Wobbrock et al. “Ability-based design: Concept, principles and examples”. In: *ACM Transactions on Accessible Computing (TACCESS)* 3.3 (2011), pp. 1–27.
- [19] Ziqian Zhou Dr. “Empathy in education: A critical review”. In: *International Journal for the Scholarship of Teaching and Learning* 16.3 (2022), p. 2.

# Evolving Cybersecurity Education: An Analysis of the GenCyber Teacher Academy's Progression from 2022 to 2023 and Beyond\*

Elizabeth A. Radday<sup>1</sup>, Mehdi Mekni<sup>2</sup>, Liberty Page<sup>2</sup>,  
Ardiana Sula<sup>2</sup>, Laura Brown<sup>2</sup>

<sup>1</sup>EdAdvance

{radday}@edadvance.org

<sup>2</sup>Connecticut Institute of Technology  
University of New Haven  
West Haven, CT 06516

{mmekni, asula, lpage, lbrown}@newhaven.edu

## Abstract

The GenCyber Teacher Academy (GTA) stands as a pioneering professional development initiative, empowering Connecticut's high school educators in diverse STEM fields to explore and integrate cybersecurity concepts into their teaching. The inaugural 2022 edition facilitated inquiry-based learning and collaborative discourse on GenCyber Cybersecurity Concepts. However, program evaluation uncovered areas for curriculum enhancement. This paper delineates the evaluation process, curriculum revisions, and their implementation outcomes. Findings demonstrate that the revised 2023 GTA fostered improved teacher engagement with modules, enhancing their ability to integrate cybersecurity principles while prioritizing online safety. Notably, the revised GTA fortified the sustainable GenCyber Teacher Academy Teaching and Learning Community, bolstering a network of educators and practitioners destined to collectively mold Connecticut's cybersecurity landscape.

---

\*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

Connecticut's educational landscape reveals a conspicuous gap in comprehensive cybersecurity education, particularly within high school curricula [1]. Despite strides in mandating Computer Science and Cybersecurity courses, the implementation faces hurdles [2]. The state's educational initiatives lack robust support structures, especially concerning professional development programs for educators[3]. This insufficiency results in marginal incorporation of cybersecurity concepts within classroom teachings, leaving students ill-prepared for the evolving digital landscape [4]. To address this critical deficit, there's an imperative for transformative initiatives empowering high school STEM educators. This paper outlines how the GenCyber Teacher Academy (GTA) [5] program evaluation process, curriculum updates, and their execution resulted in notable outcomes. The results show that the 2023 revised GTA effectively boosted teacher involvement in modules, thereby strengthening their capacity to integrate cybersecurity principles and prioritize online safety.

The remainder of this paper is organized as follows; The sections start with a background of Cybersecurity Education in Connecticut 2, followed by a description of the GTA's structure and activities in Section 3. Next, Section 4 details the implementation of the GTA program in the summer of 2022. Section 5 delves into the curriculum design and learning outcome assessment techniques employed in evaluating and revising the new GTA. Section 6, presents the results stemming from the implementation of the suggested recommendations in GTA 2023. Finally, Section 7 concludes the document, discussing the study's implications and outlining potential avenues for future work in the realm of cybersecurity education.

## 2 Cybersecurity Education in Connecticut

Connecticut recognized the importance of cybersecurity education with the introduction of the *Position Statement on Computer Science Education* by the *Board of Education*. Although a 2015 mandate required high schools to offer Computer Science (CS) and Cybersecurity courses, it lacked funding for essential professional development and curriculum support [2, 3]. This led to ongoing challenges, marginalizing cybersecurity education statewide [6]. High schools in Connecticut face difficulties in delivering cybersecurity education, particularly in districts already struggling with teacher recruitment and diversity [1, 4]. Termed "Opportunity Districts," these areas serve urban populations historically underrepresented in education [7]. Urgent efforts are needed to enhance these schools' capacity to provide valuable cybersecurity education to diverse student groups. Additionally, the state's teacher workforce lacks diversity, with



only 9.6% educators of color compared to over 45% students of color [1]. Addressing this diversity gap is crucial for creating inclusive environments where all students, including minorities, can thrive.

### 3 GenCyber Teacher Academy Structure

The recent study, described in [5], provides a comprehensive description of the GenCyber Teacher Academy (GTA) which aims to address the growing need for skilled cybersecurity professionals. GTA is Connecticut’s pioneering program training high school teachers in cybersecurity and online safety. The GTA program goals are multifaceted: 1) Developing a professional development curriculum, including culturally responsive teaching for cybersecurity education; 2) Crafting and validating cybersecurity lesson plans and related teaching materials; 3) Establishing a sustainable teaching community shaping the state’s cybersecurity future. This intensive, learner-centered program focuses on the GenCyber Cybersecurity Concepts Framework, engaging STEM educators in activities including lectures, labs, and lesson planning. The GTA curriculum is fully supportive of the GenCyber Cybersecurity Concepts as detailed in Table 1. It covers Network Fundamentals, Python Programming, Cybersecurity Ethics, Cryptography, and Social Engineering. This curriculum aligns with the GenCyber cybersecurity principles [8]. GTA offers daily Cybersecurity Seminars with industry experts and follows up with GenCyber Teacher Academy Learning Community (GTALC), providing ongoing support, mentoring, and professional development for participating teachers.

Table 1: GTA curriculum mapping of the GenCyber Cybersecurity Concepts

|                         | Networking | Python | Cyber Ethics | Cryptography | Social Engineering |
|-------------------------|------------|--------|--------------|--------------|--------------------|
| Defense in Depth        | ✓          |        | ✓            | ✓            |                    |
| Integrity               | ✓          | ✓      |              | ✓            |                    |
| Think Like an Adversary |            |        | ✓            |              | ✓                  |
| Confidentiality         |            | ✓      | ✓            |              |                    |
| Availability            | ✓          |        | ✓            |              | ✓                  |
| Keep it Simple          |            | ✓      |              | ✓            | ✓                  |

### 4 GenCyber Teacher Academy Implementation

During the summer of 2022, the first GTA took place at The University of New Haven in Connecticut. The program was developed and implemented by a diverse leadership team: a Program Director, a Lead Instructor, a K-12 pedagogy expert, two Instructors, and two Teaching Assistants. Twenty-five participants, twelve men, and thirteen women, participated in the program and were selected from a pool of 78 Connecticut high school STEM teachers. The

summer camp program ran daily from 8:00 AM to 5:00 PM for five days in August 2022. Before the intensive week-long experience, teachers completed approximately eight hours of pre-program work through Google Classroom. Post-camp the teacher group met an additional four times, once per month from September through December, for three hours at a time.

The GTA program focused on five different modules and dedicated one day of camp to each module and pre-camp work was assigned for each topic. These modules were: Cybersecurity Awareness; Python Programming and Scripting; Cryptography; Network Fundamentals; and Social Engineering. Understanding computer networks is crucial for cybersecurity professionals because it enables them to understand how data is transmitted and identify potential security threats, such as network attacks and intrusions. Python is a popular programming language that is widely used in the cybersecurity industry for writing scripts and automating various cybersecurity tasks. Knowledge of cryptography is necessary to understand how encryption algorithms work and their application to secure data communications and data transmissions, and verify that files have not been tampered with. Cybersecurity awareness training helps individuals and organizations identify and avoid potential security threats, such as phishing attacks, social engineering, and malware. Social engineering is a technique used by attackers to trick individuals into revealing sensitive information.

Participants completed a pre-camp survey to determine their prior knowledge and skills with topics in cybersecurity. After the 2022 GTA, teachers completed post-program surveys about the knowledge and skills they gained, as well as their perceptions of the modules and overall GTA camp experience.

## **5 Curriculum Design and Learning Outcomes Assessment**

The assessment of the participating teachers' learning is an essential means of demonstrating each participant has met the goals of the GTA program and identifying areas for improvement in the proposed curriculum. The GTA assessment plan is a three-tier structure that includes: formative, interim, and summative assessments.

### **5.1 GTA Formative Assessment**

Formative assessment occurs in the short term with prompt feedback from instructors. Examples of activities supporting formative assessment include self-assessment quizzes, essay assignments, and discussion forums in the pre and post-outreach phases. During the summer program, warm-up and wrap-up sessions are used daily. These sessions improve learners' retention of covered concepts and highlight the relationship with the new modules. Moreover,

reflection sessions scheduled at the end of each day of the summer program allow for engagement with learners through discussions facilitated by the lead instructor and the K12 pedagogy expert.

## **5.2 GTA Interim Assessment**

The interim assessment allows learners to demonstrate an understanding of cybersecurity-related material and concepts. Each module includes a set of hands-on laboratory exercises, homework assignments, and group-based project implementation. The prompt feedback from instructors helps recognize gaps in instruction and participants' learning. In addition, the participating teachers engage in lesson plan design, development, and validation during the summer program. Participants are expected to create a series of several lesson plans that allow them to incorporate their new learning and bring it back to their students. Feedback from the lead instructor and the K12 pedagogy specialist helps improve their course lesson plans and increase the success of their implementation.

## **5.3 GTA Summative Assessment**

The summative assessment is performed by the GTA team, upon the completion of the summer program, to identify strengths and weaknesses of the proposed curriculum and potential future improvements. Examples of summative assessment include the presentation of the produced lesson plans created by the participants during the summer program and refined in the post-outreach program supported by our GTALC events.

## **5.4 GTA 2022 Evaluation and Recommendation**

The GTA integrated assessment plan aims to build participants' confidence to teach cybersecurity in high schools. Based on the survey data from 2022, one key theme emerged that was crucial to improving the 2023 GTA academy. As depicted in Figure 1, the Networking module was rated the lowest of the five modules at the camp in several dimensions. Feedback from the students made it clear that the material was very complex and difficult to understand. Both anecdotally and in surveys the participants reported that they were the least confident in teaching networking to their students even after they participated in the instructional module at camp as displayed in Figure 2. Participant responses indicated that the material was too in-depth for novice teachers and was far too complicated to bring back to their classrooms. It was determined that the networking module would need a complete overhaul for the summer of 2023.

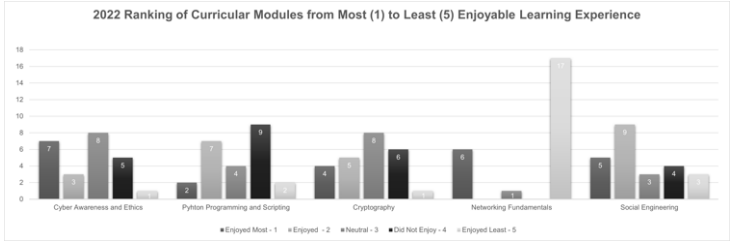


Figure 1: GTA 2022 participants ranking of the curricular modules.

Additionally, participants enjoyed the activities that were hands-on and interactive and they felt the most valuable parts of the camp were doing activities that they could bring directly back into their high school classrooms and implement. For example, the teachers enjoyed using Scytale to encode and decode messages. They enjoyed creating funny and witty messages for their peers to read aloud. All teachers were given two 3D-printed Scytales in 2 different sizes and the 3D print files so they could make their own sets at school. Teachers knew they could bring this back to school and use it as an introduction to cryptography, the same way they had done the activity in the camp. A goal for the 2023 camp was to send teachers home with as many ready-to-adapt lessons and activities that they could use in their high school classrooms as possible. Based on these recommendations, the GTA leadership

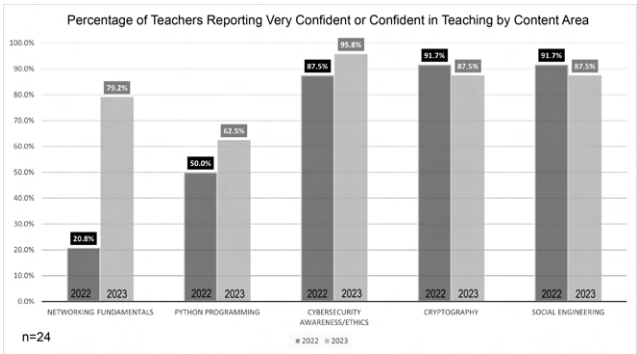


Figure 2: Evolution of participants’ confidence in teaching the curricular modules.

team aimed to overhaul the Networking module and make minor changes or additions to the other four modules to ensure that the participants were not only learning the concepts but also had a way to transfer that knowledge back

to their students in engaging and practical ways.

## 6 Results of the GTA Program Revision

The summer 2023 GTA program had 24 participants; 13 men and 11 women and were selected from a pool of 46 applicants. The camp expanded its reach beyond Connecticut and also had participants from New York and Massachusetts. A critical role added to the instructional team was the GTA Ambassador (GA). This person was a participant in the 2022 GTA camp and was selected based on her experience and enthusiasm for not only the subject material but also for her passion for teaching students and sharing her knowledge with her colleagues. Adding the GA to the team meant that the voices of the teachers were represented in the planning and execution of the camp experience. She was able to speak from the perspective of the classroom teacher, which was invaluable while planning the modules and activities.

### 6.1 Networking Module

For the 2023 GTA Camp, the team’s highest priority was to improve the networking module. Understanding the underlying basics of networking is critical to understanding how cyber attacks occur and what makes networks vulnerable to malicious acts. With the help of a new instructor and the GA, the module was completely rewritten to be more hands-on and understandable (See Figure 2). The GA and an instructor decided to use the “Internet of Strings” series of activities to help the participants understand the basics of how a network works.

#### 6.1.1 Internet of String Activity

Adapted from the work done by Hernandez et al. [9], the “Internet of Strings” (IoS) is a creative and engaging model that simplifies computer networking concepts using the analogy of strings and interconnected nodes. In this approach, data links are represented as strings, and devices become nodes in the network. By using relatable objects and scenarios, high school students can grasp complex networking principles more easily. This concept can be introduced in a classroom setting to foster curiosity and understanding of the fundamental principles that underpin modern computer networks. IoS aligns with *Integrity* and *Availability* GenCyber Cybersecurity Principles.

**Lesson Objectives** Once this module concludes, participants will find themselves equipped to navigate the intricacies of the Internet of Strings (IoS) more comprehensively. They’ll embark on a journey that begins with unraveling the

concept of IoS, drawing parallels to conventional computer networks to grasp how data flows within this innovative framework. Participants will delve into the essence of data transmission within IoS, akin to the way packets navigate through real networks. The significance of protocols governing communication rules will become clearer, illuminating the critical role they play within IoS. As they explore the terrain, nodes will emerge as pivotal players, handling and receiving strings, thereby shedding light on fundamental data processing concepts. In-depth discussions will ensue on the trifecta of data reliability, security, and network scalability within the IoS, illuminating their paramount importance. Finally, identifying challenges and brainstorming potential solutions to minimize string latency will wrap up this immersive learning experience.

**Classroom Activities** Participants engage in a series of dynamic challenges that mirror the intricate workings of computer networks. In the *String Relay Race*, groups craft strings symbolizing messages and pass them swiftly between participants, mirroring the crucial speed and accuracy required in data transmission. Similarly, the *String Maze* forms a web of interconnected nodes, guiding participants through a labyrinth while illustrating the complexities of routing decisions in effective data transmission.

The *Protocol Creation* activity invites participants to conceptualize their own "*String Protocol*," showcasing the significance of standardized rules in facilitating seamless communication across diverse devices. With the String Decoder tool, participants decode encoded messages, drawing parallels to data processing and reception within computer networks. This exercise highlights the critical role of decoding in understanding hidden messages, akin to data processing mechanisms.

Moreover, the *String Security Challenge* plunges participants into a cryptography test, where they encrypt and decrypt strings using basic encryption techniques. This exploration emphasizes the vital role of data security in safeguarding sensitive information during transmission. Lastly, the String Relay Olympics present a relay race that evolves in complexity, mirroring the challenges and solutions associated with network scalability as teams expand and interconnections grow. Through these engaging activities, participants delve into the multifaceted layers of networking concepts, experiencing firsthand the intricacies and significance of data transmission, routing decisions, protocol establishment, data decoding, security measures, and network scalability.

By employing the "*Internet of Strings*" model, the teacher participants gained a comprehensive understanding of networking fundamentals in a fun and accessible way. This hands-on approach allowed participants to relate abstract concepts to real-world scenarios, fostering a deeper appreciation for the critical role computer networks play in our digital lives. The IoS model, with its

simple yet effective analogy, can spark interest and curiosity in computer networking and serve as a stepping stone for those interested in pursuing careers in technology and computer science.



Figure 3: (a) Rock Paper Parity model, (b) Rock Paper Parity implementation, and (c) lock-picking activities

### 6.1.2 Rock Paper Parity Activity

Rock Paper Parity is an engaging educational activity that transforms the concepts of binary digits (bits) and parity prediction into an interactive game (See Figure 3a). Participants pair up, assuming roles—one as the 'bit' and the other as the 'parity predictor.' When the parity prediction is incorrect, the 'bit' transitions to find a new partner within a group that still has a parity predictor (See Figure 3b). However, upon making the right parity call, the group absorbs additional 'bits,' expanding its size and continuing the game. This dynamic encourages participants to understand the implications of correct and incorrect parity predictions while fostering collaboration and movement within the group setting. At the end of the GTA camp, participants rated this module as their favorite. Participants were offered all the supplies and accompanying lesson plans and materials to be able to bring this lesson directly back to the students. They were very enthusiastic about using this with their high school students.

### 6.2 Social Engineering Module

At the 2022 GTA camp, participants were very interested in the topic of social engineering. Given the huge role social media plays in the life of most teenagers, they believed it was not just important, but critical, that students understand how people with malicious intent can use different techniques such as phishing and baiting to obtain personal information that can be used in harmful and destructive ways. Given the interest in this topic because of its applicability to high school students, the team wanted to ensure that participants had enough

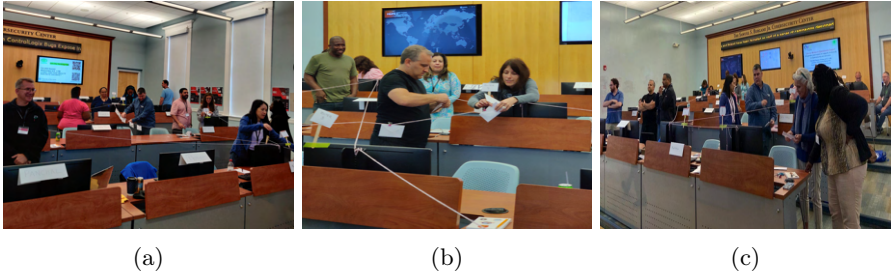


Figure 4: Internet of String activities in classrooms

information and ideas to bring back to their classrooms. In 2023, in addition to the lecture and PowerPoint presentations, the GTA instructional team created small, laminated visual cards that explain the different types of cognitive hacks that can be used to trick people into sharing their personally identifiable information. Each card also has a real-life scenario that exemplifies the tactic and can promote student discussion. All teachers were given a set of cards that could be used with their classes.

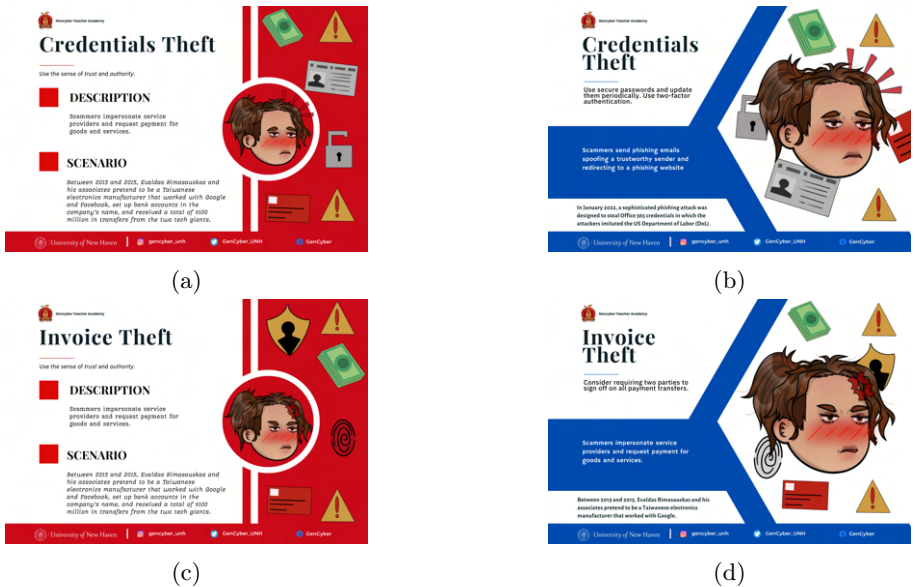


Figure 5: Samples of Social Engineering Cards (Red for Attacker (a & c) and Blue (b & d) for Defender scenarios



### 6.3 Cybersecurity Concepts Module

Throughout the week the six cybersecurity concepts were called upon frequently. By the end of the week, all students could easily name and define all six principles and apply them to a variety of cybersecurity concepts and scenarios. For the 2023 GTA camp, one of the TAs with experience in graphic design, created six posters for each of the cybersecurity concepts illustrated in Figure 6. In addition to the concept and its definition, an image was used to further illustrate the concept. For example, for the concept “Defense in Depth,” there is an image with a castle that shows that it is protected by strong walls, a drawbridge, and a moat - multiple layers of security to get inside. These posters proved helpful throughout the week for participants to refer to as they were learning. All participants were provided with digital PDF versions of the posters so they could create their own for their classroom walls.



Figure 6: Samples of media depicting the GenCyber Cybersecurity Concepts

## 6.4 Cryptography Module

The 2022 module on cryptography was successful overall. Participants enjoyed the day and the opportunity to try out different types of ciphers and enjoyed the opportunity to play with a set of locks for a lock-picking activity (See Figure 3c). This year the participants again enjoyed the module, writing each other messages using scytales, and different historical ciphers, and ending with lock picking. Each participant was given a set of locks and a lock-picking tool to use in their classroom. Teachers also worked together to think about other items they could buy cheaply to use as a scytale without having to 3D print them. They suggested using large novelty pencils or other plastic shapes they found on different websites.

## 6.5 Other Changes

The GTA program emphasizes ongoing professional growth. The instructors advocate for continual interactions among participants and mentors beyond the summer program. The GTALC holds monthly meetings during the fall, providing mentoring, coaching, and a platform for high school teachers to exchange experiences, seek advice, access resources, and enhance skills in cybersecurity-based teaching methods. This inclusive space focuses on inquiry-based pedagogy, especially for underrepresented minority (URM) and female students. The GTALC, led by our team and supported by collaborators such as the CT State Department of Education, EdAdvance, Computer Science Teachers' Association, educators, and a selection of guest speakers as depicted in Figure 2, conducts these monthly sessions from September to December, totaling fourteen hours of post-program engagement. In 2022, the five-day camp took place in a room with 5 pods of 5 desks, each with a desktop computer. Participants enjoyed the opportunity to be in groups throughout the week, however, the room was very crowded and left very little space to maneuver. The setup was static and inflexible and some participants wished the groups were mixed up throughout the week.

Table 2: Overview of GTA Guest Speakers

| Position                    | Institution                               |
|-----------------------------|---|
| Director Cybersecurity      | SullivanCotter                            |
| Project Director - CS-PLAN  | Sacred Heart University                   |
| Vice President              | Computer Science Teacher Association      |
| Chief Diversity Officer     | University of New Haven                   |
| Director of Policy Research | New England Board of Higher Education     |
| Education Consultant        | Connecticut State Department of Education |

For 2023 the camp was held in the most high-tech classroom space on the

UNewHaven campus. The room had amphitheater-type seating with three long rows of connected tables, each one rising a bit higher than the one in front of it. The room was equipped with desktop computers and a large screen in front of the room. There was ample room for moving around, but again the static nature of the space meant that it was more difficult to work in groups. Given that participants need access to computers throughout the week and not all of them have their own devices, GTA must provide access to computers. Additionally, strong WiFi is needed for everyone to be able to reliably use the internet. Finally, a space that comfortably holds 25 adult participants plus an additional 5-7 instructors and faculty is not easy to come by - especially when other camps for high school students, college students, and other adults are taking place simultaneously. Overall the new space was a positive change, but the program has yet to find that perfect spot for holding the camp.

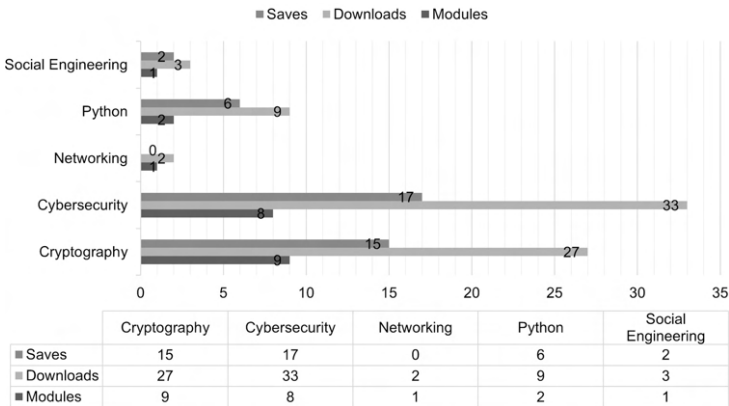


Figure 7: GTA 2023 curricular modules published on CLARK [10]

## 7 Conclusion and Future Work

The Python Programming module has been a persistent challenge for participants in 2022 and 2023. The difficulty spans the spectrum: some found it too basic, while others, particularly STEM teachers less acquainted with coding, faced difficulties. Notably, adept participants supported their peers, patiently guiding them through increasingly complex coding tasks. The GTA team, cognizant of this hurdle, is resolute in enhancing the module for the 2024 program.

Under consideration for improvement includes scrapping the Python module in favor of a different topic in cybersecurity. The rationale aligns with the limited application of Python skills among non-CS teachers and the redun-

dancy for experienced coders. Proposing an alternative, a prospective module on Artificial Intelligence in cybersecurity emerges as a timely, relevant solution. Moreover, integrating Python basics, and leveraging ChatGPT’s coding capabilities, could illuminate its role in AI-centric cybersecurity[11, 12, 13].

Another major improvement of the GTA program targets the adopted data strategy. Through systematic data collection, management, and analysis, we aim to delve deeply into the impact and efficacy of our training program. This approach not only gauges success and areas needing enhancement but also upholds the highest standards of data ethics and compliance. By utilizing insights garnered from this strategy, the GTA adapts to evolving educational needs and technological advancements. Its components include collecting participant data, assessing skill development, monitoring classroom implementation, ensuring secure data management, conducting regular analyses, refining programs based on insights, employing technology for deeper insights, and prioritizing compliance and ethical data practices. This comprehensive strategy fortifies our commitment to strengthening cybersecurity education.

The 2023 GTA underwent substantial improvements. Recognizing teachers’ affinity for cybersecurity knowledge applicable in their classrooms, the leadership team pivoted towards more hands-on activities and resource provisions. This strategic shift fostered a positive learning environment, empowering participants to confidently wield their newfound expertise in their educational settings. Participants of GTA 2023 designed and published more than twenty cybersecurity curriculum modules, focusing on Cybersecurity and Cryptography topics. These modules were approved and made available on CLARK [10], a repository offering free, high-quality cybersecurity lessons (See Figure 7).

The GTALC unites educators across cohorts to collaborate and enhance technology education, especially in cybersecurity and digital literacy (See Table 2). By facilitating resource sharing and providing current industry insights, it empowers educators to engage students in cybersecurity, aiming to cultivate their interest and prepare them as proficient digital citizens. The GTALC goes beyond traditional knowledge exchange, nurturing curiosity, innovation, and a holistic understanding of digital safety and security.

## References

- [1] T. Gais, B. Backstrom, J. Frank, and A. Wagner, “The state of the connecticut teacher workforce.” *Nelson A. Rockefeller Institute of Government*, 2019.
- [2] Connecticut Examiner, “Computer science education expanding in k-12,” 2019. [Online]. Available: <https://ctexaminer.com/2019/12/04/computer-science-education-expanding-in-k-12/>

- [3] Senate and H. of Representatives in General Assembly, “Substitute senate bill no. 962, public act no. 15-94,” <https://www.cga.ct.gov/2015/ACT/pa/pdf/2015PA-00094-R00SB-00962-PA.pdf>, 2015.
- [4] Connecticut State Department Of Education, “Connecticut teacher shortage areas report 2020–2021,” 2020. [Online]. Available: <https://portal.ct.gov/-/media/SDE/Performance/Research-Library/ConnecticutTeacherShortage-Areas-Report-2020-21.pdf?la=en>
- [5] L. D. Page, M. Mekni, and E. A. Radday, “Incorporating cybersecurity concepts in connecticut’s high school stem education,” *J. Comput. Sci. Coll.*, vol. 38, no. 8, p. 173–187, apr 2023.
- [6] K. Dell, N. Nestoriak, and J. Marlar, “Assessing the impact of new technologies on the labor market: Key constructs, gaps, and data collection strategies for the bureau of labor statistics,” 2020.
- [7] Connecticut State Department of Education, “Opportunity district,” 2021. [Online]. Available: <https://portal.ct.gov/-/media/SDE/Alliance-Districts/Opportunity-District.pdf?la=en>
- [8] “Cybersecurity first principles,” 2022. [Online]. Available: [https://mlhale.github.io/nebraska-gencyber-modules/intro\\_to\\_first\\_principles/README/](https://mlhale.github.io/nebraska-gencyber-modules/intro_to_first_principles/README/)
- [9] J. Hernandez, X. Qu, X. Yuan, and J. Xu, “Engaging middle and high school students in cybersecurity through summer camps,” in *2020 ASEE Southeastern Annual Section Conference*, 2020.
- [10] “Clark center for global engagement,” <https://clark.center/home>, accessed: March 28, 2024.
- [11] M. Mekni, S. Atilho, B. Greenfield, B. Placzek, and M. Nassar, “Real-time smart parking integration in intelligent transportation systems (its),” in *Proceedings of the Future Technologies Conference*. Springer, 2023, pp. 212–236.
- [12] C. R. Barone IV, M. Mekni, and M. Nassar, “Gargoyle guard: Enhancing cybersecurity with artificial intelligence techniques,” in *2023 3rd Intelligent Cybersecurity Conference (ICSC)*. IEEE, 2023, pp. 127–132.
- [13] M. Vondráček, I. Baggili, P. Casey, and M. Mekni, “Rise of the meta-verse’s immersive virtual reality malware and the man-in-the-room attack & defenses,” *Computers & Security*, vol. 127, p. 102923, 2023.

# Can ChatGPT Pass a CS1 Python Course?\*

James S. Sharpe, Ryan E. Dougherty, and Sarah J. Smith  
Department of Electrical Engineering & Computer Science  
United States Military Academy  
West Point, NY 10996

{james.sharpe, ryan.dougherty, sarah.smith}@westpoint.edu

## Abstract

In this paper we determine whether an LLM–ChatGPT in this case—can successfully complete the assignments in our CS1 course as if it were a “real” student. Our study contains a two-stage approach, involving reprompts to the LLM in the cases of either not successfully completing the assignment, or using concepts that are more advanced than are taught in our course. We find that LLMs can in fact can either perfectly solve, or almost perfectly solve, every assignment in our CS1 course.

## 1 Introduction

The emergence and rapid development of ChatGPT has inspired much research regarding its abilities and potential capabilities in education, especially where it applies to programming education. Topics explored include ChatGPT’s ability to accomplish programming tasks in different contexts as well as its ability to aid in student learning. In this paper we determine how well ChatGPT can solve assignments within our CS1 course.

## 2 Related Work

Yilmaz and Yilmaz [5] collected data through questionnaire questions from 41 undergraduate students who were told to use ChatGPT in coursework for an

---

\*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.

object-oriented programming course at a state university in Turkey. Coursework came in the form of individual weekly laboratory assignments and students were encouraged to use ChatGPT to complete these assignments. At the end of the course, students evaluated its benefits and limitations. Students responded to the survey that ChatGPT's greatest benefits were its ability to "respond to questions quickly and effectively," save time doing coursework, and to help with "complex problems." The limitations included some concerns about programmers becoming "lazy" as well as having "occupational anxiety" students felt because of ChatGPT's effectiveness, but also disadvantages such as ChatGPT's not giving correct answers every time, not having sufficient information, and not giving the desired answers at times. The responses saying that ChatGPT helps with solving complex problems somewhat conflict with those saying that ChatGPT often gives incorrect answers. From these differing sentiments, ChatGPT may have difficulty answering questions correctly in specific contexts, or when dealing with specific types of problems, rather than having more trouble with more complex problems. Our research is sufficiently different than this study because it evaluates ChatGPT's direct abilities with the course, rather than involving students.

Piccolo et al. [4] evaluated ChatGPT's ability to solve programming exercises in a life science context with the greater interest of investigating ChatGPT's potential to assist life scientists by writing code or by assisting them to write code. They evaluated ChatGPT using 184 Python programming exercises with different bioinformatics contexts and separated the results based on ChatGPT's ability to solve the problem within a given number of attempts. ChatGPT solved 139 (75.5%) of the exercises in one attempt, and they gave additional guidance to ChatGPT to help it understand exercise requirements, yielding 179 (97.3%) correctly solved exercises within 7 attempts. The unsolved exercises all came from later parts of the course, requiring multiple techniques to be used simultaneously. They also reported results indicating that ChatGPT was more likely to have difficulty with problems with biological contexts specifically. From these results, ChatGPT appears to have more difficulty both when using multiple techniques at once with longer solutions and in more specialized contexts, though the study only deals with biological contexts specifically. Our research is similar to this project but is distinct because of its overall scope of an entire course as well as involving a more general introductory coding context.

Tung Phung et al. [3] compared the performance of ChatGPT and GPT-4 with human tutors in introductory programming education and debugging scenarios with the intent of enhancing programming education. The programming scenarios included tasks such as providing hints and feedback as well as fixing buggy problems. They also tested the models on introductory Python

problems with relatively short solutions. The wide range of evaluations in this study showed a significantly higher performance of GPT-4 over ChatGPT, both of which were still outperformed by human tutors. The largest difference between GPT-4 and human tutors involved problems with grading feedback and creating problems for students to practice skills, with tutors performing much higher. From these results, for large language models to serve in a tutoring role, some improvements must be made in areas of feedback and assignment creation, to match high demands for educational content. This study is sufficiently different from our research because we evaluate ChatGPT’s performance in a course within the role of a student, rather than that of a tutor.

Geng et al. [1] investigated how well ChatGPT performs in an introductory functional language programming course. The considered coursework included both exam questions and standard programming assignments, with topics including higher-order functions, recursion, and pattern matching. ChatGPT was tested in two phases: “unassisted” and “assisted.” In the unassisted phase, ChatGPT was given only the problem prompts as they were given to students, whereas in the assisted phase, prompt engineering techniques were used to attempt to get the correct answer. Without assistance, ChatGPT scored perfectly on 16 of 31 assignment problems, resulting in a rank of 220 out of 314 human students. This rank improved with prompt engineering, placing it at 155 out of 314 with a B- in the course. Our research distinguishes itself from this study by instead focusing on an intro imperative programming language course, and does not contain those three topics.

Kiesler and Schiffner [2] investigated ChatGPT’s performance in introductory programming exercises with the intent of examining the model’s potential in the programming education field. To evaluate the model, 72 Python tasks were taken from the site CodingBat<sup>1</sup>. ChatGPT and GPT-4 were both evaluated on their ability to accomplish the programming tasks as well as their ability to explain them textually. Feedback was provided on which test cases were failed for each problem when code failed. Although the authors note that ChatGPT was less receptive to feedback than GPT-4, both models performed significantly well, above 94% for each. One explanation is that the authors used feedback and additional prompts, indicating that large language models could be very helpful for education when used with prompt engineering techniques and understanding of model limitations. Our research is different in that we evaluated ChatGPT for our course both with and without feedback. Further, we used the Python programming language, which provides less of a difficulty curve in introducing “advanced” features, and thus ChatGPT’s outputs may be more likely to include such features.

---

<sup>1</sup><https://codingbat.com/java>



| Problem Set | Tested Topics                               |
|-------------|---|
| 1           | Functions                                   |
| 2           | Basic Selection and Repetition              |
| 3           | Advanced Selection, Repetition, and Strings |
| 4           | File I/O                                    |
| 5           | Exceptions, Lists, and Tuples               |
| 6           | Sequence Functions and Methods              |
| 7           | Classes, Nested Lists, and Mutability       |
| 8           | Dictionaries                                |
| 9           | Collections                                 |

Table 1: Topics for each problem set in our course.

### 3 Institutional and Course Context

The CY300 course at the United States Military Academy, a small liberal-arts college in the Northeastern United States, is required for all students majoring in the Computer Science and Cyber Science majors as well as for non-majors taking a 3-course elective sequence. There are approximately 1100 new students enrolled each year at the United States Military Academy, and approximately 100 of them choose to be a CS or CY major. Each class section has at most 19 students. All students take a fundamentals of computing course, a prerequisite to CY300. The CY300 course covers introductory Python programming, with nine problem sets featured as main homework assignments, building on concepts learned in class. Each problem set is focused on a different area of Python programming, as shown in Table 1.

The entire course is 1000 points, with 300 of them assigned to the problem sets. Problem sets 1-3 are 20 points each and Problem Sets 4-9 are 40 points each. Fifteen small in-class quizzes are 4 points each. Two midterm tests are 125 points each. A course project accounts for 140 points. The remaining 250 points come from a final exam. Midterm tests and the final exam consist of both a coding portion as well as more conceptual knowledge. The course project requires students to work in pairs and find a data set online. Their responsibility is to code a user interface to manipulate the data set in different ways.

The nine problem sets require functions to be coded, which are checked against test cases by a custom-built autograder. The grading for these problem sets is almost entirely based upon these test case results. Some problem sets require partial manual grading, such as Problem Set 5, which involves exceptions. Students have the ability to view their test case results, including expected and received outputs, before submitting the assignment for all

problem sets except for the last two. For these, only the overall score is shown. Problem sets may be submitted as many times as needed before the assignment deadline.

Problem set questions identify required inputs to functions and expected outputs returned by the function. The questions also identify any assumptions that can be made about inputs and outputs. The following is an example from question 5 of Problem Set 5.

Define a function named `get_names` that has one parameter, a string that contains multiple email addresses separated by semi-colons. The function should return a list of tuples containing the user's (`lastname`, `firstname`). Assume all email addresses are valid in the format

`first.last@university.edu` or `first.last3@university.edu`.

If the email address contains a number you can assume that it is one digit between the last name and the @ symbol.

```
>>> get_names('james.bond@university.edu')
[('Bond', 'James')]
>>> get_names('james.bond3@university.edu')
[('Bond', 'James')]
>>> sorted(get_names('james.bond@university.edu;taylor.
    swift@university.edu;michael.jordan3@university.edu'))
[('Bond', 'James'), ('Jordan', 'Michael'), ('Swift', 'Taylor
    ')]
```

## 4 Method

We seek to determine how well ChatGPT can score on these nine problem sets. For each problem set, we created a new ChatGPT session, and used two approaches sequentially to generate a correct answer. Problem sets are re-made from year to year, so the problem sets used in this study are unlikely to be part of ChatGPT's training data. GPT-3.5 was used for this study.

### 4.1 Approach 1: Verbatim Problem Sets/Results

The flowchart for Approach 1 in our study is shown in Figure 1. The first prompts given to ChatGPT involved providing only the verbatim problem set instructions document given to students. Since GPT-3.5 only allows plain-text inputs for prompts, instructions that were contained within images or tables were not provided in the prompt to ChatGPT, even those that consisted mainly

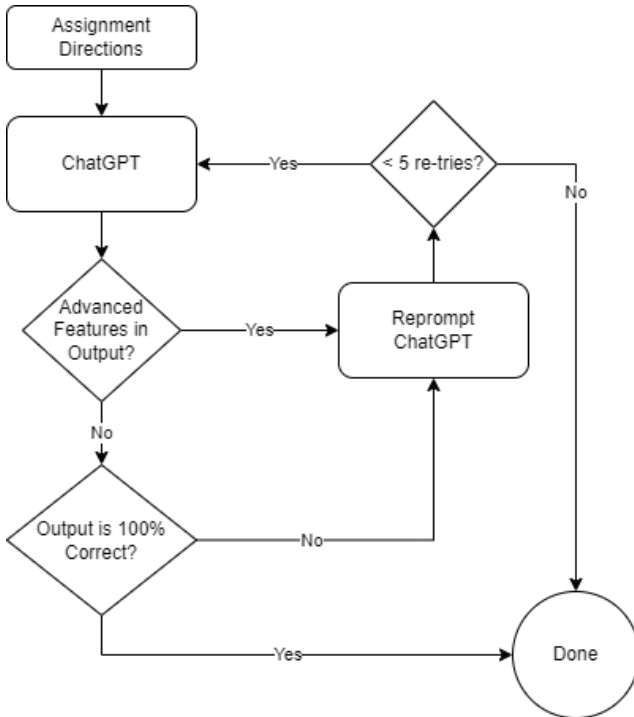


Figure 1: Flowchart for Approach 1 in our Study.

of text. Information that was not relevant to the problems themselves, such as submission guidelines, was omitted. In the prompts, ChatGPT was advised to include all of the problem set’s functions in one block of code, with any annotations as comments as necessary for the problem set. This emulates the submission standard for students in CY300. If ChatGPT did not provide a code output in this format, we re-asked ChatGPT until it did so.

After ChatGPT generated a properly formatted output, the code was submitted to the autograder. If any test cases failed, the generated results for those specific test cases were provided back to ChatGPT, with one sentence to indicate which problem the test case results were and instructions to return a corrected product. The test cases that were returned consisted of the failed function name, its input values, the expected output, and the received output.

This process was repeated for subsequent outputs until ChatGPT either generated a solution that produced a perfect score or until progress halted, either through reaching five attempts or through a lack of a response to the corrective prompts. In times where the corrective prompts gave no change in the code, an additional line was added to the next prompt, pointing out the lack of change. After five repeated failures or a lack of response to corrections, a second approach was taken, as described in Section 4.2. If the feedback resulted in a perfect score, no further action was taken with that problem set.

## 4.2 Approach 2: Unrestricted Feedback

In the second approach, corrective prompts had more specific guidance given, ranging from less to more direct assistance. We began in the same session as the first approach, with corrective prompts beginning after ChatGPT’s highest scoring output from it. The initial prompts were less specific, such as interpreting errors and incorrect outputs in a more diagnostic fashion, rather than just giving the correct output. If this failed to cause improvement, prompts advanced to more direct descriptions of what mistakes ChatGPT was making, describing the actual code that it outputted, rather than the values it returned. From this, if the code still failed to satisfy all test cases, corrective prompts would move on to suggest specific code to be used. If tests still failed beyond this, ChatGPT will have failed to satisfy all test cases required for that function of the problem set.

## 5 Results

Table 3 shows ChatGPT’s scores when only Approach 1 was used as well as when Approach 2 was allowed, if it was necessary. Also listed are the advanced Python features used in each problem set’s solution. For the purposes of this study, “advanced” refers to those concepts that students are not expected to

| Problem Set | Advanced Python Features          |
|-------------|-----------------------------------|
| 1           | None                              |
| 2           | None                              |
| 3           | list comprehensions, dictionaries |
| 4           | list comprehensions               |
| 5           | regular expressions (initially)   |
| 6           | Counter (initially)               |
| 7           | list comprehensions               |
| 8           | None                              |
| 9           | list comprehensions               |

Table 2: Advanced features used in the program outputs of ChatGPT for each of the problem sets.

| Problem Set | Approach | Approach 1 Score | Final Score |
|-------------|----------|------------------|-------------|
| 1           | 1        | 100%             | 100%        |
| 2           | 2        | 95%              | 100%        |
| 3           | 2        | 95%              | 100%        |
| 4           | 2        | 80%              | 80%         |
| 5           | 2        | 65%              | 100%        |
| 6           | 1        | 100%             | 100%        |
| 7           | 1        | 100%             | 100%        |
| 8           | 2        | 72.5%            | 100%        |
| 9           | 1        | 100%             | 100%        |

Table 3: Approach used, the score achieved by ChatGPT for Approach 1, and the final score for each of the problem sets.

know by that specific point in the course. In the case of very advanced features, such as those not found in the textbook for the course, ChatGPT was directed to regenerate its output without the use of those concepts.

For most problem sets, ChatGPT outputted code that resulted in a 100% score, with the exception of Problem Set 4, which was scored 80%. List comprehensions were the most often used advanced feature. Since they are demonstrated in the textbook, ChatGPT was not directed to regenerate outputs because of its use of list comprehensions.

We provide two illustrative examples. The first is in Figure 2, where ChatGPT achieved a 100% score on the first attempt; in this case, for Problem Set 5. The figure contains four items: (1) a screenshot of the original problem set directions, (2) the prompt to ChatGPT, (3) the relevant code portion for that problem produced by ChatGPT, and (4) the instructor solution for this

problem. As is evidently clear, the ChatGPT code is isomorphic to the instructor solution, apart from one variable name change. For this problem set, all of the questions were written in plaintext, thus were able to be copied and pasted. Note that ChatGPT did not generate a docstring; although it was not explicitly prompted to do so, the prompt did contain a valid docstring.

The second example is in Figure 3, which contains Problem Set 5's first question. Here, we provide the (1) original directions, (2) the prompt to ChatGPT, (3) the relevant code ChatGPT produces, and (4) the instructor solution. However, (3) contains usage of the regular expression Python module, which is not taught in our course. Therefore we (5) re-prompted ChatGPT to solve the same problem without the module, which it was able to do.

## 6 Discussion

### 6.1 Problem Set 1: Functions

In Problem Set 1, ChatGPT gained a score of 100% with just one initial prompt, without any advanced features used.

### 6.2 Problem Set 2: Basic Selection and Repetition

ChatGPT scored 95% of the points for Problem Set 2 using Approach 1 only. Problems that ChatGPT encountered difficulties involved finding multiples just before a specific number. For one problem, the function was supposed to return the multiple of an input that was just before the integer 50000. The other problem required a multiple of 57 that was just before the input value. For both functions, values that were returned were off by one multiple from the correct answer. Most of these mistakes were corrected by providing test case results repeatedly, but ChatGPT was unable to make progress using this method with one specific test case in the second function, with complications caused by negative inputs. Approach 2 began by pointing out the issues caused by negative numbers, and asking ChatGPT to account for them. After this failed, ChatGPT was asked to attempt a different technique, as its attempt to use an if else statement to account for the negative numbers was failing. Another difficulty occurred when the input itself was not a multiple of 57. For this, specific guidance was required in order to fix the mistake, to include suggesting a solution. After corrections in Approach 2, ChatGPT achieved a score of 100%.

2. Define a function named `check_names` with two parameters. The first is a list of names you will search through and the second will be a name you are looking for. If the name is in the list return True otherwise `raise` a value error with the message: `'Name' not found in list`.

Note: This problem is specifically designed to assess your ability to `raise` an error. Normally, when we raise an error we also want to catch that error and help our program handle that error gracefully.

```
...
>>> name_list = ['Frodo', 'Merry', 'Sam', 'Pippin', 'Aragorn',
' Boromir', 'Legolas', 'Gimli', 'Gandalf']
>>> check_names(name_list, 'Frodo')
True
>>> check_names(name_list, 'Harry')
Traceback (most recent call last):
...
ValueError: 'Harry' not found in list.
...
```

2. Define a function named `check_names` with two parameters. The first is a list of names you will search through and the second will be a name you are looking for. If the name is in the list return True otherwise a value error with the message: `'Name' not found in list`.

Note: This problem is specifically designed to assess your ability to raise an error. Normally, when we raise an error we also want to catch that error and help our program handle that error gracefully.

```
"
>>> name_list = ['Frodo', 'Merry', 'Sam', 'Pippin', 'Aragorn', 'Boromir', 'Legolas', 'Gimli', 'Gandalf']
>>> check_names(name_list, 'Frodo')
True
>>> check_names(name_list, 'Harry')
Traceback (most recent call last):
...
ValueError: 'Harry' not found in list.
"
```

```
# Problem 2
def check_names(name_list, name):
    if name in name_list:
        return True
    else:
        raise ValueError(f"'{name}' not found in list.")
```

```
# Problem 2
def check_names(names, name):
    """
    >>> name_list = ['Frodo', 'Merry', 'Sam', 'Pippin', 'Aragorn', 'Boromir', 'Legolas', 'Gimli', 'Gandalf']
    >>> check_names(name_list, 'Frodo')
    True
    >>> check_names(name_list, 'Harry')
    Traceback (most recent call last):
    ...
    ValueError: 'Harry' not found in list.
    """

    if name in names:
        return True
    else:
        raise ValueError(f"'{name}' not found in list.")
```

Figure 2: Example of ChatGPT's achieving 100% on the first attempt; this is Problem Set 5's second question. The boxes in order are the original problem set directions, the prompt to ChatGPT, the relevant response by ChatGPT, and the instructor solution.

1. Define a function named `create_word_list` that has two parameters, a string and an integer. The string represents the name of a file. If the file exists, the function should read the contents of the file and identify all words of length `n`, where `n` is the integer passed to the function. Return a list of the words of length `n`. If the file does not exist return the value `None`. If there are no words in that file of length `n`, return an empty list. (+2 pts for properly implementing error handling)

```

problem1.txt - Notepad
File Edit Format View Help
red
orange
yellow
green
blue
purple

```

```

'''
>>> create_word_list('file_does_not_exist.txt', 3)
>>> create_word_list('problem1.txt', 9)
[]
>>> create_word_list('problem1.txt', 3)
['red']
>>> sorted(create_word_list('problem1.txt', 6))
['orange', 'purple', 'yellow']
'''

```

```

# Problem 1
def create_word_list(filename, n):
    try:
        with open(filename, 'r') as file:
            words = re.findall(r'\b\w{0}\b' % n, file.read())
            return words
    except FileNotFoundError:
        return None

```

Anonymous  
I don't understand the re module. Could you do the problems without using it?

ChatGPT  
Certainly! You can solve the problems without using the 're' module. Here's the updated code for each problem without using regular expressions:

```

python
# Problem 1
def create_word_list(filename, n):
    try:
        with open(filename, 'r') as file:
            words = [word.strip() for line in file.readlines() for word in line.split()
                    if len(word) == n]
            return words
    except FileNotFoundError:
        return None

```

```

# Problem 1
def create_word_list(file_name, n):
    """
    >>> create_word_list('file_does_not_exist.txt', 3)
    >>> create_word_list('problem1.txt', 9)
    []
    >>> create_word_list('problem1.txt', 3)
    ['red']
    >>> sorted(create_word_list('problem1.txt', 6))
    ['orange', 'purple', 'yellow']
    """
    list_of_words = []
    try:
        with open(file_name, "r") as file:
            words = file.read().split()
    except FileNotFoundError:
        return None
    for word in words:
        if len(word) == n:
            list_of_words.append(word)

```

Figure 3: Example of ChatGPT's solving Problem Set 5's Question 1 with an "advanced" feature: the regular expression module. We include the re-prompt to ChatGPT to solve the same question without this feature.



### 6.3 Problem Set 3: Advanced Selection, Repetition, Strings

ChatGPT achieved a score of 95% using Approach 1 only for Problem Set 3. The only function that ChatGPT had trouble creating was a variant of Rock, Paper, Scissors under the lens of selecting a local restaurant to get food from. ChatGPT was able to improve itself to only fail one test case, but with repeated corrective prompts after that, it would revert to failing more cases. The functions theme seemed to be confusing ChatGPT, causing it to create a function that did not perfectly resemble Rock, Paper, Scissors. Approach 2 took to more specifically describing how the function should look as Rock, Paper, Scissors with different names. Then, some additional prompts were required to correct some combinations and to alter returned values to use the correct wording in order to have correct results in test cases. After corrections in Approach 2, ChatGPT achieved a score of 100%.

### 6.4 Problem Set 4: File I/O

ChatGPT achieved a score of 80% using both approaches with Problem Set 4. The failed test cases were for two functions. In one, the function was to return a flag found in an input file. In the other, the function was supposed to write a feedback file for a homework assignment in which students were to write the first and last letters of a given word, given an input file with the student answers. In both functions, ChatGPT's code returned errors as a result of an incorrect interpretation of the input file format. Students completing these problems have access to sample input files, allowing them to better code the functions. ChatGPT not having access to these sample files likely caused the errors. With repeated corrective prompts, errors were fixed, but ChatGPT continued to struggle with correctly identifying the flags in the input files and with reading from and writing to files for the second function. With the second approach, ChatGPT failed to make changes to its code despite varied prompts, thus failing to improve on its score.

### 6.5 Problem Set 5: Exceptions, Lists, Tuples

ChatGPT achieved a score of 65% on Problem Set 5 using Approach 1. ChatGPT failed test cases on two functions. The first function is given a list and an integer. If the integer is a valid index in the list, it returns the value at that index. If not, it returns the error given by Python. Initially, ChatGPT encountered difficulty because it returned the error message as a string, rather than as the error object itself, as required by the problem. This problem was only improved upon with Approach 2, as ChatGPT only changed its answer correctly when this problem was pointed out in words, rather than in test case results.

The other function ChatGPT failed to achieve a 100% on immediately required getting names from email addresses in a specific format. ChatGPT failed test cases where the email addresses included numbers with names. For this function, ChatGPT initially used regular expressions, a concept far beyond the scope of the course. As part of corrections in Approach 2, ChatGPT was asked to create the function without the use of the `re` module. In doing so, ChatGPT improved the function to 100% without additional feedback. ChatGPT's resulting score for Approach 2 was a 100% for Problem Set 5.

## 6.6 Problem Set 6: Sequence Functions and Methods

In Problem Set 6, ChatGPT eventually scored 100%. For one function, students are required to determine whether or not a word can be made with a given set of letters. ChatGPT created the function initially using `Counter` from the `collections` module, creating a correct function. However, because this concept is beyond the scope of the course, ChatGPT was asked to generate an output without using `Counter`. After a re-prompt, ChatGPT then revised the function with a less advanced technique, resulting in the same score.

## 6.7 Problem Set 7: Classes, Nested Lists, Mutability

In Problem Set 7, ChatGPT gained a score of 100% with just one initial prompt, with the most advanced technique used being list comprehensions, which can be found in the course textbook in assigned readings, thus being within the scope of the course.

## 6.8 Problem Set 8: Dictionaries

ChatGPT scored 72% for Approach 1 in Problem Set 8. It had difficulty with two functions. The first function required the use of a key to decode a message, given a file with the message and a dictionary as the key. ChatGPT encountered difficulty when messages used both uppercase and lowercase letters. Corrective prompts initially failed to cause any change in the function output. With descriptive corrections in Approach 2, ChatGPT was able to discern between cases for letters, and maintain the case in the decoded message.

The other function that for which ChatGPT failed to produce a fully correct output required a dictionary to be created from a given `csv` file containing UFO sighting statistics by state. Because the file's format was not viewable by ChatGPT, initial solutions caused errors, which weren't able to be corrected through Approach 1 alone. This function was corrected with a single prompt using Approach 2, simply by describing which column in the `csv` file contained

the information required for the solution. As a result, ChatGPT improved to 100% after Approach 2.

## 6.9 Problem Set 9: Collections

ChatGPT achieved a score of 100% in Problem Set 9 through the use of multiple prompts containing test cases. This problem set’s test cases are unique in that they do not specify which problem is being tested by name, but this can be determined through the tests themselves. Most of the failed test cases for this problem set were results of file input/output difficulties. ChatGPT encountered errors with several functions because it did not include code to skip a header line in the instructor-provided `csv` files. After one corrective prompt, this issue was fixed.

The remaining failed test cases were for one function: given two text files, determine which file has more unique words. ChatGPT failed test cases due to case sensitivity, which was again corrected after one more prompt with test cases only.

## 7 Future Work

This paper examines ChatGPT 3.5’s ability to generate correct code for various introductory programming assignments. GPT-4 could be examined similarly in future work to identify areas where it improves upon 3.5. Of particular interest in the context of this paper, the number of additional prompts required could be expected to decrease. Also, since ChatGPT gives different outputs even with the same prompt, we want to analyze all of these outputs, especially with regard to the advanced Python features list in Table 2.

One line of future research we are currently investigating is making the analysis more granular in the following sense. We want to model three types of typical students: (1) a student who aces everything in a CS1 course; (2) a student who is “average” in that they achieve a B-/C+ average in a CS1 course; and (3) a student who fails a CS1 course. From the perspective of these three student models, Table 2 is interpreted differently; an instructor would likely be fine with model 1’s using those features, but less so for the other two models.

## 8 Conclusion

In this paper we conducted a study to determine how well ChatGPT can solve the problem sets in a CS1 course that uses the Python programming language. Although it did not completely solve all problem sets, it more than adequately

passed the course after at most five re-prompts. We then discussed our findings and what reasonable future work should entail with LLMs.

## References

- [1] Chuqin Geng et al. *Can ChatGPT Pass An Introductory Level Functional Language Programming Course?* arXiv:2305.02230 [cs]. May 2023. URL: <http://arxiv.org/abs/2305.02230> (visited on 09/22/2023).
- [2] Natalie Kiesler and Daniel Schiffner. *Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments.* arXiv:2308.08572 [cs]. Aug. 2023. URL: <http://arxiv.org/abs/2308.08572> (visited on 09/22/2023).
- [3] Tung Phung et al. "Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors". en. In: *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2*. Chicago IL USA: ACM, Aug. 2023, pp. 41–42. ISBN: 978-1-4503-9975-3. DOI: 10.1145/3568812.3603476. URL: <https://dl.acm.org/doi/10.1145/3568812.3603476> (visited on 09/22/2023).
- [4] Stephen R Piccolo et al. *Many bioinformatics programming tasks can be automated with ChatGPT.* en. Mar. 2023. URL: <https://arxiv.org/abs/2303.13528>.
- [5] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. "Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning". In: *Computers in Human Behavior: Artificial Humans 1.2* (Aug. 2023), p. 100005. ISSN: 2949-8821. DOI: 10.1016/j.chbah.2023.100005. URL: <https://www.sciencedirect.com/science/article/pii/S2949882123000051> (visited on 09/21/2023).

# Infusing Computational Thinking into a Computer Science Gateway Course\*

Younes Benkarroum, Mohammad Q. Azhar  
Computer Information Systems  
Borough of Manhattan Community College  
New York, NY 10007  
`{ybenkarroum,mazhar}@bmcc.cuny.edu`

## Abstract

Computational thinking (CT) stands as a universal problem-solving approach applicable across diverse disciplines, transcending the domain of computer science. It embodies the mental process of structuring a problem to enable a computational solution feasible for both humans and machines. This methodology involves dissecting problems into smaller parts that are easier to understand and solve. This study delineates a meticulously designed series of CT activities within an introductory computer science course and explores their profound impact on student engagement and problem-solving proficiency. Our findings underscore the pivotal role of hands-on CT practice in augmenting students' ability to decompose problems, recognize patterns, and abstract complexities, and employ algorithms effectively. Notably, this infusion of CT not only cultivates theoretical understanding but also bridges the gap between conceptual knowledge and real-world application through the use of computational tools like Python programming. As CT continues to emerge as a cornerstone skill in diverse domains, this research presents compelling evidence advocating for its integration into introductory courses, laying a robust foundation for students to navigate the evolving technological landscape with enhanced problem-solving capabilities.

---

\*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

Promoting computational thinking (CT) has emerged as a pivotal educational objective spanning both STEM and non-STEM domains. CT stands as an inclusive problem-solving approach, extending beyond computer science to become as fundamental as core competencies like literacy and numeracy, as articulated by Professor Wing [10]. Understanding the scope and constraints of CT, as underscored by Lu and Fletcher [4], holds significant relevance, even for individuals outside the STEM sphere, contributing substantially to professional and everyday contexts.

Computational thinking encapsulates the cognitive process of framing problems to accommodate solutions executable by humans, machines, or their amalgamation [11]. This methodology involves breaking problems down into separate parts (decomposition), looking for similarities or common differences (pattern recognition), filtering out information that is not necessary to solve the problems (abstraction), and developing step-by-step instructions for solving problems (algorithms) [9].

Between Fall 2021 and Fall 2023, as part of our NSF-funded project “Bridging the Gap: Designing a Technology Learning Community Integrating Computational Thinking to Improve STEM Engagement across Disciplines”, we curated a series of five computational activities aimed at honing students’ skills in practicing and perfecting their computational thinking abilities. The students who participated in the activities were from the introductory course CSC 101 - Principles in Information Technology. This article details the approach and outcomes observed among a total of 142 CSC 101 students in addressing these computational thinking activities.

# 2 Related Work

Previous studies have underscored the synergy between creative thinking and computational thinking (CT), advocating for their combined integration into educational frameworks. Miller et al. [5] demonstrated the efficacy of introducing creative thinking exercises within introductory computer science courses, a concept echoed in our approach. The introduction of our CT activities not only mirrors this integration but also provides a practical platform where students engage in creative problem-solving scenarios that revolve around computational challenges.

Moreover, the study by the same research team [7] that introduced Computational Creativity Exercises (CCE) into an introductory computer science course tailored for engineering students aligns closely with our methodologies. Just as their outcomes highlighted an enhanced grasp of fundamental CT prin-

ciples, our meticulously designed CT exercises aimed to not only introduce core CT concepts but also engage students in practical problem-solving tasks, observing similar positive impacts on students’ understanding and application of CT methodologies.

Furthermore, the work by Shell et al. [6], emphasizing the correlation between self-regulation, creative competence, and CT skills, aligns with our approach of providing a scaffolded learning environment. Our activities aimed to scaffold the development of CT skills by progressively guiding students through the elements of decomposition, pattern recognition, abstraction, and algorithm design, promoting a structured approach to problem-solving similar to that advocated by prior research.

Lu and Fletcher [4] proposed that the mental models and patterns fostered by computational thinking create an environment conducive to teaching higher-level computational processes and abstraction. Csizmadia et al. [3] presented a guide for integrating computational thinking into curricula, emphasizing its fundamental role in education. Swaid [8] promoted the universal relevance of computational thinking as an underlying mindset in STEM projects and education discussions. Castro et al. [2] analyzed the impact of a first-year engineering course on the acquisition of computational thinking across various student profiles.

In essence, these studies collectively underscore the significance of merging creative thinking with CT, scaffolding learning experiences, and reinforcing theoretical concepts through practical engagement — principles that resonate strongly with the integrative CT activities outlined in this study.

### 3 CT Exercise Design

Our activities introduced students to the problem of finding the shortest path between multiple cities on a road map. The activities we designed aim to cultivate computational thinking, which is foundational for problem-solving in computer science. By breaking down complex problems into manageable parts and devising algorithms, students develop crucial problem-solving strategies applicable across various domains within computer science.

When traveling from one place to another, there are several factors to consider; unless the intent is to visit some points and landmarks in a specific order, people are often interested in the most efficient way to get somewhere. In our activities, students used CT to experiment with different ways of creating paths between multiple points to effectively travel through cities in a region. This problem is also known as the Traveling Salesman Problem (TSP) [1] where the salesman must take the shortest path that passes through each city exactly once and returns back to the start. Thus, the problem statement of our exer-

cise is: Given a list of cities in South Africa, the goal is to reduce the cost for a vendor whose duty is to distribute the COVID-19 vaccines in those cities. The problem is therefore to find the shortest possible route that starts from Cape Town and visits each city exactly once.

As stated earlier in the introduction, CT is a problem-solving process that includes four elements: decomposition, pattern recognition, abstraction, and algorithms. In the following, we focus on one CT element at a time.



Figure 1: Elements of Computational Thinking Process

### 3.1 Decomposition

Breaking problems down into smaller parts can make complex challenges more manageable. This allows other elements of CT to be applied more effectively to complex challenges. Applying the decomposition element to our problem leads to the following sub-tasks:

1. Given the list of cities, determine their geographic coordinates (i.e., latitude and longitude) to find their exact positions on the map.
2. Find the distances between each pair of cities in the list.
3. Find the shortest path that passes through each city exactly once.

The first two sub-tasks are quite straightforward and easy to accomplish using the available online maps and their geographic tools; but the last sub-task is the main section of our activity. In the following, we will only describe how the remaining CT elements can be applied to this section.

### 3.2 Pattern Recognition

Recognizing if a pattern exists and determining its sequence can simplify the solution. For the task at hand, we noticed that the problem can be solved recursively as follows.

Let the letter O (Origin) be the label of the departure city (Cape Town). If we only need to visit two cities (A and B), then the strategy we use to



determine the shortest path will choose either the path O-A-B or the path O-B-A. Thus, the number of possible paths to choose from is 2.

If the number of cities to visit is three (A, B, and C), then the first city to be visited can be either O-A, O-B or O-C, then the remaining two cities must be visited. Note that from the previous case, the strategy knows how to determine the shortest path when visiting two cities, so the number of possible paths in this case is  $3 \times 2 = 6$ .

In general, if the number of cities to visit is  $n$ , then the number of possible paths to choose from is  $n \times P(n-1)$ , where  $P(n-1)$  is the number of possible paths when visiting  $n-1$  cities.

### 3.3 Abstraction

Stepping back from the specific details of a given problem and focusing on the big picture allows us to create a more generic solution. Applying this CT element in our case requires us to analyze the problem to leave out unnecessary information such as the mode of transportation chosen, the purpose of the trip, the time needed to visit all cities, whether the vendor is traveling with coworkers, etc. Once done, we can start brainstorming a solution to the problem. The important details to focus on are the list of cities, their exact locations, and the distances between them.

### 3.4 Algorithms

An algorithm is a step-by-step strategy for solving a problem. It can be written in plain language, with flowcharts, or pseudocode. In our activity, students from CSC 101 and BUS 104 tried four different algorithms and compared their outputs. All algorithms have been applied to the list of the given cities from South Africa; the starting point was always Cape Town, and then the algorithms produced routes that visit all other cities. The algorithms we considered in this activity are listed below.

1. **Longitude sorting:** The task is to sort the cities by their longitudes, then to visit them all in ascending order of this geographic coordinate. By doing so, the path traveled will visit all cities from left to right (from the farthest city in the west to the farthest city in the east). The total route distance is the sum of the distances between each pair of cities on the route.
2. **Latitude sorting:** The task now is to sort the cities by their latitudes, and then visit them all in ascending order of that coordinate. In doing so, the path traveled will visit all cities from the farthest city in the south to the farthest city in the north.

3. **Nearest neighbor:** The nearest neighbor algorithm starts at Cape Town and connects to the nearest unvisited city. It repeats until every city has been visited.
4. **Greedy algorithm:** The greedy algorithm is an iterative algorithm that builds a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. In our activity, the edges are the roads between the pairs of cities. All possible edges are sorted by distance in ascending order; then at each iteration, we add the shortest edge which will not make a city with more than 2 edges, nor create a cycle.

## 4 Results and Discussion

It's important to highlight that due to the staggered offering of activities across various semesters and days, not all students engaged in every activity. While some students participated in all activities, others took part in only specific ones. Overall, among the various activities offered, a total of 142 unique students participated in at least one. Figure 2 illustrates the distribution per semester of the students who participated in the CT activities, while Table 1 and Figure 3 display the counts and rates of students who successfully completed each of the four algorithms.

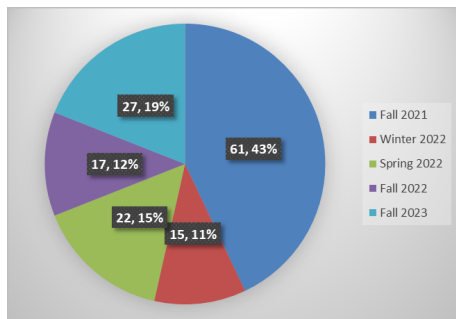


Figure 2: Distribution per Semester of the Participants

From the description of each of the algorithms above, we can observe that the initial two activities present relatively straightforward tasks. The initial low success rate for the Longitude Sorting (75.3%) might indicate a lack of clarity in instructions or unfamiliarity with the activity format. This could be attributed to it being the first task. The lesson plan regarding the Longitude Sorting activity is appended at the end of this article. The high success rate for

Table 1: Number of Participants and Success Rate by Activity

| Algorithm         | Participants | Successful Completion | Success Rate |
|-------------------|--------------|-----------------------|--------------|
| Longitude sorting | 77           | 58                    | 75.3%        |
| Latitude sorting  | 69           | 61                    | 88.4%        |
| Nearest neighbor  | 85           | 66                    | 77.6%        |
| Greedy algorithm  | 41           | 23                    | 56.1%        |

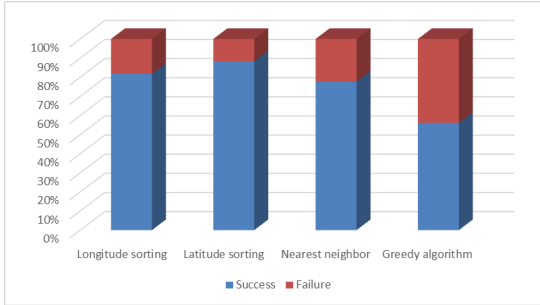


Figure 3: Success Rate by Activity

the Latitude Sorting (88.4%) indicates that once students understood the format and requirements (thanks to the prior task), they performed significantly better. Familiarity likely contributed to this high success rate. The Nearest Neighbor task had a relatively moderate success rate (77.6%) which seems reasonable considering the complexity of the algorithm. The Greedy Algorithm task had the lowest success rate (56.1%) as anticipated due to its difficulty. Being a challenging algorithm, it's expected that a smaller percentage of students would successfully complete this task, especially in an introductory course.

The varying success rates signify the students' level of engagement and understanding of CT concepts. Higher success rates, such as those observed in tasks following simpler algorithms after initial exposure, indicate a growing familiarity and grasp of CT principles. These observed rates offer pedagogical insights, guiding educators in designing future CT activities. Understanding which CT elements pose challenges or foster better comprehension can aid in refining teaching methodologies and curriculum design to optimize student learning experiences.

It's worth noting that the participating students did not utilize programming skills to determine the shortest routes for the four explored algorithms. Instead, they manually executed each algorithm's steps and solely employed the Google Sheet *SUM* function to compute the total distance for each path. As

depicted in Table 2, there has been a progressive enhancement in the distance of the shortest path generated by these algorithms.

Table 2: Distance of Shortest Path by Algorithm

| Algorithm         | Distance |
|-------------------|----------|
| Longitude sorting | 4,941 km |
| Latitude sorting  | 4,004 km |
| Nearest neighbor  | 3,846 km |
| Greedy algorithm  | 3,586 km |

Before engaging in the computational thinking activity, students were prompted with questions such as, *'Imagine planning a road trip with multiple stops; how would you determine the route?'* Interestingly, some students independently devised steps similar to those employed in the nearest neighbor algorithm, unaware of its formal designation. Notably, many perceived this approach as highly efficient for solving the traveling salesman problem. Subsequently, upon exposure to the greedy algorithm, students exhibited fascination, particularly upon realizing the considerable distance saved by this algorithm. Their reaction reflected an enthusiastic appreciation for the algorithm's optimization capabilities.

## 5 Brute Force Algorithm

To evaluate whether the CT activity generated the optimal route, students were tasked with employing the brute force method. This algorithm represents the initial and most straightforward strategy when confronted with a problem. In technical terms, it involves exhaustively considering all available possibilities to resolve a problem. In our case, the brute force algorithm would generate every conceivable path among the cities, subsequently computing the distance for each path and selecting the shortest one. This approach seems to be simple and achievable. But practically this is not the case unless you get help from a computing device. The number of all possible paths in our situation is exceedingly large; we have already demonstrated that the number of possible paths to choose from is  $P(n) = n \times P(n-1)$ . This sequence can be simplified as follows:

$$P(n) = n(n - 1)(n - 2) \dots 1 = n!$$

With eight cities to visit in our activity, the total count of potential paths amounts to  $8!$  (i.e., 40,320). Manually calculating all these paths is impractical for students. Hence, they employed a computer program to generate and evaluate all paths, subsequently selecting the one with the shortest distance. To

address this challenge, students were provided with a concise Python program utilizing a *for-loop* to systematically generate and display the various paths for visiting three cities (labeled as B, C, and D) from the initial city A and computing their respective distances. The Python program listing is provided below, and Figure 5 showcases the program's output.

```
import itertools

Cities = "ABCD"

Distances = [[0, 40, 20, 90],
              [40, 0, 30, 70],
              [20, 30, 0, 55],
              [90, 70, 55, 0]]

AllPaths = list(itertools.permutations(sorted(Cities[1:])))

for p in AllPaths:
    path = list(p)
    path.insert(0,"A")
    path = tuple(path)
    print(path, end = ' ')

    sum = 0
    for x in range(len(path) - 1):
        sum += Distances[Cities.find(path[x])][Cities.find(path[x + 1])]

    print("distance =", sum)

print("\nNumber of paths =", len(AllPaths))
```

Figure 4: Brute Force Python Program

```
('A', 'B', 'C', 'D') distance = 125
('A', 'B', 'D', 'C') distance = 165
('A', 'C', 'B', 'D') distance = 120
('A', 'C', 'D', 'B') distance = 145
('A', 'D', 'B', 'C') distance = 190
('A', 'D', 'C', 'B') distance = 175

Number of paths = 6
```

Figure 5: Brute Force Python Program

Building upon the above Python program, the students' initial task involved modifying the code to accommodate the traversal of eight cities in South Africa, considering the actual distances between each pair of cities.

Due to the large number of paths presented on the program's output (40,320), students encountered difficulty identifying the path with the shortest distance. To overcome this challenge, students were instructed to enhance the Python program by integrating a decision-making statement (if-statement) and the necessary variables to capture the smallest distance and its corresponding path, and subsequently displaying this information on the screen. Among the 91 participating students, 76 successfully completed this task, resulting in a 83.5% success rate. Remarkably, the shortest path distance recorded aligned precisely with the distance computed using the greedy algorithm. This alignment intensified the students' fascination with the efficiency and accuracy of the greedy algorithm, solidifying their appreciation for its problem-solving prowess.

It's important to note that while the optimal shortest path distance aligned with the output of the greedy algorithm in this specific instance, this outcome may not necessarily hold true for all scenarios. The apparent optimality observed could be coincidental, and further investigation is warranted to determine the general applicability of the greedy algorithm across various problem instances.

## 6 Conclusion

In summary, the infusion of computational thinking (CT) into the Computer Science Gateway Course was instrumental in enhancing problem-solving skills among students. Through a series of meticulously designed activities, participants were not only introduced to the core principles of CT but also engaged in practical problem-solving scenarios. The findings from this study highlight the progressive enhancement in students' comprehension and application of CT methodologies across diverse algorithms, with varying complexities.

The analysis of students' performance revealed insightful patterns, showcasing the impact of familiarity and task complexity on success rates. Notably, the success rates varied across algorithms, shedding light on the challenges students encountered and conquered. Despite the complexities, students demonstrated a commendable grasp of CT elements, showcasing their ability to decompose problems, recognize patterns, abstract details, and employ algorithms effectively.

Moreover, the integration of CT wasn't solely confined to theoretical understanding; rather, it required practical implementation, necessitating creativity and critical thinking. The utilization of computational tools like Python programming served as a bridge between theoretical concepts and real-world problem-solving, enhancing students' ability to navigate complex scenarios.

In conclusion, this study has significant implications for educational strategies, emphasizing the importance of hands-on engagement and practical ap-

plications in computational thinking education. As CT continues to emerge as a foundational skill in various domains, its integration into introductory courses lays a robust foundation for students, equipping them with essential problem-solving abilities crucial in today’s evolving technological landscape.

As a part of future work, the authors aim to conduct a comparative study between students enrolled in CS introductory courses where computational thinking (CT) was infused and those where it was not. This comparative analysis will focus on evaluating the performance of students when confronted with novel problems demanding solution methodologies. It is hypothesized that students exposed to CT methodologies will showcase enhanced problem-solving capabilities in tackling new challenges. This comparative analysis holds the potential to offer valuable insights into the tangible impact of integrating CT into introductory CS courses. The aim is to discern the influence of CT infusion on students’ adaptability and efficacy in addressing novel problems, ultimately gauging the efficacy of CT as a foundational educational approach.

## Acknowledgements

This work was supported by the U.S. National Science Foundation under Award #2122690. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

## References

- [1] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton, New Jersey: Princeton University Press, 2006.
- [2] Laura Melissa Cruz Castro et al. “Analyzing Students’ Computational Thinking Practices in a First-Year Engineering Course”. In: *IEEE Access* 9 (2021), pp. 33041–33050.
- [3] Andrew Csizmadia et al. “Computational Thinking - a Guide for Teachers”. In: *Computing at School* (2015), p. 18.
- [4] James J. Lu and George H.L. Fletcher. “Thinking About Computational Thinking”. In: *Proceedings of the 40th ACM technical symposium on Computer science education* (2009), pp. 260–264.
- [5] L. Dee Miller et al. “Improving Learning of Computational Thinking Using Creative Thinking Exercises in CS-1 Computer Science Courses”. In: *43rd ASEE/IEEE Frontiers in Education Conference Proceeding* (2013), pp. 1426–1432.

- [6] Duane F. Shell et al. “Associations of Students’ Creativity, Motivation, and Self-Regulation with Learning and Achievement in College Computer Science Courses”. In: *IEEE Frontiers in Education Conference* (2013).
- [7] Duane F. Shell et al. “Improving Learning of Computational Thinking Using Computational Creativity Exercises in a College CS1 Computer Science Course for Engineers”. In: *IEEE Frontiers in Education Conference* (2014).
- [8] Samar I. Swaid. “Bringing Computational Thinking to STEM Education”. In: *Procedia Manufacturing* 3 (2015), pp. 3657–3662.
- [9] Kristen Thorson. “Early Learning Strategies for Developing Computational Thinking Skills”. In: *Getting Smart* (2018).
- [10] Jeannette M. Wing. “Computational Thinking”. In: *Communications of the ACM* 49 (2006), pp. 33–35.
- [11] Jeannette M. Wing. “Computational Thinking: What and Why?” In: *The Link Magazine* (2010), pp. 20–23.

## Appendix – Longitude Sorting Lesson Plan

### Finding the Shortest Path Challenge – Activity 1

There are several factors to consider when traveling from one place to another. Unless the intent is to visit some points and landmarks in a specific order, people are often interested in the most efficient way to get somewhere. In this activity, students will use computational thinking to experiment with different ways of creating paths between multiple points to effectively travel through cities in a region.

#### Factors involved when choosing a route:

**Question:** If you are traveling from one place to another, how do you decide what route to take? What are the factors that influence your decision?

**Question:** Imagine you were asked to create a road trip with multiple stops, how would you decide what route to take?

#### Traveling Salesman Challenge:

Given a list of cities, their geographic coordinates, and the distances between each pair of them, the goal is to reduce the cost for a salesperson whose duty is to distribute COVID-19 vaccines in those cities. The task is therefore to find the shortest possible route that visits each city exactly once. The list we consider in this activity includes the following cities in South Africa:

- A - Cape Town



- B - Bhisho
- C - Pietermaritzburg
- D - Bloemfontein
- E - Kimberley
- F - Johannesburg
- G - Mahikeng
- H - Nelspruit
- I - Polokwane

The list of cities above along with their geographic coordinates are also included in the Shortest Path Challenge workbook that you will use in this computational thinking activity. The exact locations of the cities are shown on the map below.



The distances in kilometers (km) between each pair of cities are shown in the following matrix:

| CITY | A     | B     | C     | D     | E   | F     | G     | H     | I     |
|------|-------|-------|-------|-------|-----|-------|-------|-------|-------|
| A    | 0     | 991   | 1,558 | 1,004 | 941 | 1,398 | 1,318 | 1,743 | 1,719 |
| B    | 991   | 0     | 590   | 557   | 659 | 890   | 966   | 1,198 | 1,241 |
| C    | 1,558 | 590   | 0     | 556   | 714 | 490   | 746   | 616   | 796   |
| D    | 1,004 | 557   | 556   | 0     | 165 | 397   | 445   | 743   | 718   |
| E    | 941   | 659   | 714   | 165   | 0   | 478   | 365   | 814   | 789   |
| F    | 1,398 | 890   | 490   | 397   | 478 | 0     | 291   | 327   | 331   |
| G    | 1,318 | 966   | 746   | 445   | 365 | 291   | 0     | 632   | 545   |
| H    | 1,743 | 1,198 | 616   | 743   | 814 | 327   | 632   | 0     | 301   |
| I    | 1,719 | 1,241 | 796   | 718   | 789 | 331   | 545   | 301   | 0     |

**Note:** The above matrix is symmetric since the distance to travel from  $X$  to  $Y$  is the same as the distance to travel from  $Y$  to  $X$ .

Developing a strategy for traversing all points: In this activity, you will compare four different algorithms and choose the best one. Algorithms will be applied to all the above cities (from South Africa); the starting point is always city  $A$ , then the algorithms will produce routes that visit all the other cities. The algorithms we are considering are:

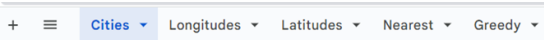
- Longitude sorting
- Latitude sorting
- Nearest neighbor
- Greedy algorithm

**Longitude sorting algorithm:**

The task is to sort the cities by their longitudes, and then to visit them all in ascending order of this geographic coordinate. By doing so, the path traveled will visit all cities from left to right (from the farthest city in the west to the farthest city in the east).

Create your local copy of the Shortest Path Challenge workbook. To do this, click on the **File** tab and then select **"Make a copy"** from the drop-down menu.

Give your local copy a name, then click **OK**.  
The workbook has five worksheets.



The first worksheet (**Cities**) includes the list of cities along with their geographic coordinates and the distance matrix.

Click on the second worksheet (**Longitudes**) that we will be using in this task. This page has the list of cities and their longitudes. Use the sort function to sort cities by longitude in ascending order. To do this, click on the arrow in column **C**, then select **"Sort sheet A -> Z"**.

|   | A     | B                | C         |
|---|-------|------------------|-----------|
| 1 | Label | Name             | Longitude |
| 2 | A     | Cape Town        | 18.4241   |
| 3 | B     | Bhisho           | 27.4411   |
| 4 | C     | Pietermaritzburg | 30.3794   |
| 5 | D     | Bloemfontein     | 26.1596   |
| 6 | E     | Kimberley        | 24.7499   |

Once sorted, fill in the second column (**City**) of the Order table in the same worksheet using the resulting order.

Use the distance matrix to fill in the third column (**Distance from Previous City**) of the **Order** table.

Once all the distances are entered, the spreadsheet will automatically calculate and display the total distance at the bottom of the **Order** table. The total distance is the sum of the distances between each pair of cities on the route.

Answer the following questions using this form.

**Question 1:** What is the order of each city in the route produced by the longitude sorting algorithm?

**Question 2:** What is the total distance of the route?

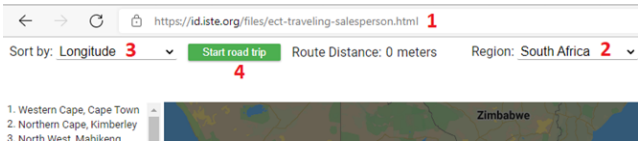
**Question 3:** Does it look like an efficient route?

**Question 4:** What is the difficulty of applying this algorithm? Use 1 for the easiest and 5 for the most difficult.

**Traveling salesperson simulator:**

Now you can use an online mapping tool to check the results of the longitude sorting algorithm. The steps are listed below.

1. Open the [Traveling salesperson simulation](#). Click **OK** in the message window that appears.
2. Use the drop-down menu in the upper-right to set the region to **South Africa**. Click **OK** again.
3. Use the "**Sort by**" drop-down menu in the upper-left to select to sort the cities by **Longitude**.
4. Press the green "**Start road trip**" button to begin the trip.



Once the road trip is complete, review the route on the map and compare the accumulated distance traveled with the total distance you found (your answer to question 2). Note that the distance returned by the online mapping tool uses the meter (m) as the unit of measure. The distance you calculated in the worksheet uses kilometers (km).

Well done! You've successfully completed the first computational thinking activity and are ready to move on to the second activity (**Latitude Sorting**).

# Engaging CS1 Students with Audio Themed Assignments\*

Matthew McQuaigue Mack Larson Philip Smith  
Sydney Melech Kalpathi Subramanian Erik Saule  
Department of Computer Science  
University of North Carolina at Charlotte  
Charlotte, NC 28223

{mmcquaig,clarson9,psmit145,smelech,krs,esaule}@charlotte.edu

## Abstract

Early computer science courses (CS1, CS2) are the cornerstone of student understanding of computer science. These courses introduce the foundational knowledge of computer science needed to understand more complex topics and to be successful in follow-on courses. It is thus important to introduce CS concepts in an engaging and easy-to-understand manner to increase student interest and retention. This paper presents a new approach to teaching the Computer Science 1 (CS1) course through our *BRIDGES* system. This approach aims to increase student engagement and improve learning outcomes by using *audio-based assignments* that they can manipulate and process audio signal information, as well as visualize and play them. We explain how to design and implement audio-based assignments and connect them to fundamental programming constructs such as variables, control flow, and simple data structures, such as arrays. These assignments encourage and engage students by using audio data they are interested in to write code, promoting problem-solving and improvements in their critical thinking skills.

---

\*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

Early computer science courses are pivotal in teaching the foundations of computer science in introducing topics that shape the future of students' academic journeys and careers. On top of foundational knowledge of computer science, students learn computational literacy, problem-solving, and critical thinking skills. As beginning courses in the field contain a lot of information, these courses can be complex and frustrating for students, and for some students, it could be the first time they are introduced to these topics. Even though progress is being made in retention rates in computer science using a variety of mechanisms and pedagogies, additional motivation and engagement is critical in early CS courses to continue to increase retention, and equip students with a strong foundation for future success. From establishing a strong base for advanced coursework to preparing students for the demands of the workforce, we believe CS1 should emerge as a transformative experience that extends far beyond the confines of the classroom to ensure that they understand the potential of computing to solve real-world problems.

Traditional programming courses often rely on text-based exercises, which can make it difficult for students to stay interested and may not cater to their diverse learning styles. In response to this challenge, our proposed curriculum includes a series of programming tasks that use audio-based stimuli to create a dynamic and multisensory learning experience. Using the *BRIDGES* system, [10, 9, 5] we have created a database of assignments for introductory computer science courses [4] and also demonstrate how they could be used in CS1/CS2. The *BRIDGES* API is available in Java, C++, and Python, and provides students with objects to load and manipulate real-world audio files in early CS courses. It also creates visualizations and plays corresponding audio data to demonstrate student-generated work while maintaining course rigor.

This paper contributes to the ongoing discourse on innovative teaching methodologies in computer science education and provides practical insights and tools for instructors seeking to enhance their students' introductory programming experience. By embracing audio-based assignments, educators can create a more inclusive and interactive learning environment, ultimately preparing better novice programmers for the challenges of the digital era. These assignments rely on the manipulation of wavefront audio information and the generation of signals to create and edit audio data at a level that is appropriate for novice programmers. The proposed assignments provide two advantages, (1) they continue to provide highly engaging assignments with a visual and audio output for CS1 students, and, (2) students' output can be customized based on different songs or around choices of their own interest. This gives more flexibility to assignments and allows students to manipulate real-world audio data that can be shared with other students outside the class.

## 2 Related Works

### 2.1 What Makes Students Engaged

In previous years, work has been done to better engage and motivate students in early CS1 courses. According to Hendlesman et al. [21] engagement can include the involvement of skills, participation, emotion, and/or performance. The methods of implying engagement in the classroom are either *content-based* or pedagogical activities. Content-based engagement focuses on making course content or activities (assignments, lectures, videos, etc) meaningful and relevant to student interests. A good resource for this type of learning engagement can be assignment repositories such as Nifty assignments [29], EngageCSEdu [26], and game-themed assignments [14]. These types of learning can tackle real-world coding challenges in collaborative teams, fostering teamwork and practical problem-solving skills. This allows for the inclusion of code reviews and critiques to encourage a culture of constructive feedback, allowing students to refine their coding practices through peer interaction. Engagement based on activities includes active learning, lab-based instruction, flipped classrooms, gamification, peer learning/coding, and multimedia content [30, 19, 22]. Gamifying by incorporating elements like coding competitions or level progression adds a competitive yet motivating dimension while using methods such as peer learning models to provide hands-on, immediate experiences that solidify theoretical concepts.

### 2.2 What CS1 Courses Typically Look Like

CS1 courses usually serve as the introductory course in computer science. This course can teach the fundamentals of programming such as variables, data types, control flow, functions, and algorithmic thinking. Algorithmic thinking can help break down complex problems into smaller, more manageable components. Some CS1 courses also introduce simple data structures in the form of lists and strings. CS1 often uses programming languages such as Java, C++, or Python. The curriculum heavily involves hands-on coding assignments that range from simple exercises and projects, foster problem-solving skills, language syntax and semantics. Exemplars of CS1 courses can be found in the ACM 2013 Curriculum guidelines [23]. Due to the wide variations in learning environments, tools, student population, and demographics, there is no one-size-fits-all approach to teaching CS1. Additionally, there are strong opinions regarding when and how object-oriented programming should be taught [6]. This is also true for discussion about IDEs [31] and languages [25] to use. Many introductory courses now incorporate graphics, GUIs, and visualizations [17, 10]. These are sometimes used as creative output in student projects, or

to illustrate key aspects of underlying objects or algorithms [13, 24, 7, 18, 32].

Furthermore, some institutions are experimenting with introductory computer science courses to now incorporate other advanced topics at a beginner level increasing multidisciplinary value. This includes games [2], robotics [12], image processing and generations [3], and statistics and data science [20, 11].

### 2.3 Existing Educational Efforts incorporating Audio Related Content in CS Courses

Past audio-based learning tools have explored the CS foundations supporting the integration of auditory learning in computer science education through manipulating audio to reinforce concepts in CS courses. Burg et al. [8] created a book and online learning supplements to make CS concepts more engaging and relevant by using sound/music applications. The authors map their curricular material to core CS courses, giving examples of programming assignments using sound/music applications that teach fundamental CS skills. Adams et al. [1] created a tool called Thread Safe Audio Library (TSAL) to explore the use of algorithm sonification - representing algorithm behavior using sound - as a pedagogical tool for computer science education. An assessment found that sonification improved students' long-term recall of the relative speeds of sorting algorithms, providing evidence that sonification aids learning. Students also rated sessions with visualizations as more engaging than non-visual sessions. TuneScope [27] enabled using sound-based activities into Snap!.

EarSketch [15] developed by Freeman et al. is a free, web-based learning environment that teaches introductory computer science through music. It provides an in-depth introduction to computer science and programming through composing, producing, and remixing music with Python and JavaScript code. Freeman et al. also found that using EarSketch showed statistically significant gains in student attitudes across 7 constructs related to computing engagement and motivation in underrepresented groups in high school [16].

## 3 A Set of Engaging and Scalable Audio Assignments

This section presents a set of assignments, all suitable for CS1/CS2 classes and leveraging the *BRIDGES* toolkit. Table 1 presents the assignments, related topics that they map to, and their engagement characteristics.

The assignments cover most of the introductory computer science class topics, including function calls, control flow, conditionals, lists, and OOP. The engagement strategies are all content-based. The most common engagement characteristic is that the assignment produces a visual and audio output that can be interpreted and played. Most of the assignments use real audio wave

| Assignment   | Topics                | Engagement  |
|--|-----------------------|---|
| Subsampling<br>Bitdepth                            | Variables, Data Repr. | Visual/Audio Output, Audio Choice   |
| Thresholding                                       | Conditionals          | Visual/Audio Output, Real Data, Audio Choice  |
| Scaffolded Player                                  | Strings, IO, Loops    | Visual/Audio Output, Real Data Analysis, Audio Choice                                     |
| Effects<br>(Mixing, Fade, Volume Change, Clamping) | 1D arrays             | Visual/Audio Output, Real Data, Real Problem, Multiple Sounds                             |
| Visualize Volume Levels                            | 2D arrays             | Visual/Audio Output, Real Data, Real Problem, Audio Choice, New Visualization             |
| Full Frequency Player                              | Functions             | Visual/Audio Output, Real Data Analysis, Audio Choice, Tool Building, Fun                 |
| Instrument Types                                   | Basic Objects         | Visual/Audio Output, Simulation, Fun, Create Unique Beats                                 |
| Basic Compression                                  | Project               | Visual/Audio Output, Uses all topics, Fun outputs, and Competitions for best compression. |

Table 1: A set of assignments using audio data to teach CS1.

data and perform a real analysis or solve a real-life problem. For all assignments, instructors have the flexibility to scaffold what information and functions to expose to the student.

The assignments are all downloadable online<sup>1</sup> and are scaffolded. Solutions are shared with instructors on request. All assignments can be done in any of the three programming languages supported by *BRIDGES*, namely, C++, Java, and Python.

### 3.1 How Do Computers Encode Sound?

All our activities rely on the simple processing of sounds in a computer. Audio signals can be represented in several ways in a computer. However, we are mostly interested in manipulating the raw audio signal, similar to what would be stored in an uncompressed wave file.

An audio clip is represented as several channels: 1 for mono, 2 for stereo, 6 for 5.1 systems. Each channel represents the movement that a membrane has to make to play the signal back. The position of the membrane is sampled at regular intervals and that information is encoded as a sampling rate. For instance, a musical CD encodes the music sampled at 44.1KHz, while telephony is often encoded at 8KHz. So a channel can be seen as a given number of samples that encode a particular clip that lasts  $\frac{\text{number of sample}}{\text{sampling\_rate}}$  seconds.

<sup>1</sup><https://bridgesuncc.github.io/>



Finally, each sample represents a displacement of the membrane compared to its resting position which takes its most positive value when the membrane should be the farthest outward from the resting position, and the most negative value when the membrane should be the most inward to the resting position. The samples can be represented in different formats. Usually, they are represented by signed integers, though the number of bits differs per quality level. For instance, music CDs were encoded at 16-bit, while telephony is often encoded in 8-bit, and the master used in music production are often encoded in 24-bit or 32-bit.

In summary, an audio signal is essentially one or multiple array(s) of integers that explicitly encode a sound wave. This representation is the one used by *BRIDGES* which lets you load a WAV file in that format and manipulate it with simple API calls. This is the format that students will use. Even though advanced audio processing will require complex mathematics (such as Fourier Transform to shift to the frequency domain), all our activities use the simpler representation which is suitable for novice programmers while enabling multiple assignments to span the content of a CS1 course.

### 3.2 How *BRIDGES* Works

The *BRIDGES* toolkit provides a simple API to create and manipulate an `AudioClip` object as a collection of samples, which are numbers, each one representing a single part of the sound wave in the clip. Currently, you can generate an `AudioClip` by setting the samples individually or by importing a WAV file.

The `AudioClip` object will also have a sample count, a channel count, a bit depth, and a sample rate. These attributes can be custom-set and retrieved through function calls from the object. The channel count is the number of “channels”, or continuous streams of samples. With multiple channels, you can play different sound waves simultaneously, which is used for things like stereo audio (a different wave for each ear).

The WAV file can be passed in as a parameter to the `AudioClip` object using a string that points to a WAV file. The `AudioClip` object will read and parse the WAV file in a form for processing for the user. The *BRIDGES* API provides a visualization function that can have an `AudioClip` attached to and sent to the server for visualization. Students can then visit the website through their account and playback the audio clip with visuals provided, as can be seen in Figure 1. Their visual clips can also be shared with other students.

The *BRIDGES* API has multiple forms of documentation and tutorials on how to manipulate and use the objects *BRIDGES* provides<sup>2</sup>. Due to the simplicity of the API and the level of scaffolding provided in the assignments,

---

<sup>2</sup><https://bridgesuncc.github.io/tutorials/AudioClip.html>

minimal effort is needed to understand and use the API even for new programmers.

## 4 Audio Based Assignments

### 4.1 Subsampling/Bitdepth

In this assignment, students will conduct audio processing by manipulating the sample rate, and bit depth of the signal through attributes of the `AudioClip` object to analyze the differences between the original and modified audio signal. Additionally, they are asked to analyze how these changes affect the file size and quality. The assignment also focuses on the impact of changing variable types, such as using 16-bit vs 32-bit integers, or floating point values on precision, and compression.

Since this assignment requires the manipulation of arrays that contain samples and 1D arrays have not been a topic yet, scaffolded code is used that is complete and takes the variables and uses them as needed. In later assignments, more and more of this code can be exposed to the student for implementation and follow-on assignments. The flexibility of what code and when it is exposed to the student is up to the instructor.

Throughout the assignment, students will focus on the understanding of audio processing while encouraging critical thinking about performance-based trade-offs of selecting certain sample rates, bit depths, and variable types. Instructors could grade for the correctness of implementation and clarity in the explanation of parameter effects. Students could also choose different sound files based on their interests to help boost engagement.

### 4.2 Thresholding

For conditionals, students can learn about thresholding to create a basic noise gate for noise reduction. To start, the students will be given the task of loading an audio file and visualizing its waveform using *BRIDGES*. After that, they will be taught to go through the audio samples and apply a threshold to identify portions of the signal that exceed a certain amplitude level. If arrays haven't been taught yet, a function can already be provided to loop through the array.

After identifying portions of the audio that fall within a certain threshold, students would use a conditional statement to modify or remove those samples. To learn more about thresholding and how it affects output quality, students should be encouraged to experiment with different threshold levels and observe how changing the conditional statements affects the audio output. They would also explore how their approach handles various types of audio signals, such as speech, music, or environmental sounds. Students can see how this form of

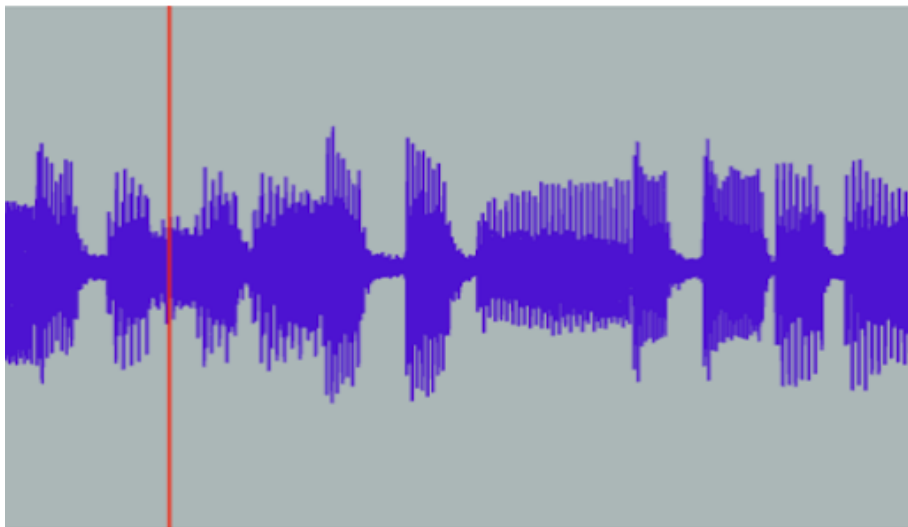


Figure 1: This is a sample audio signal visualized using *BRIDGES*. This visualization has playback buttons along with an interactive window for analyzing the signal.

noise reduction affects different types of audio. Different sound clips or songs will have different levels of noise and thus be affected more.

From this assignment, students get an understanding of how conditionals can be applied to real-world audio signals. This assignment also focuses on the importance of proper parameter selection and the impact choices have on audio quality.

### 4.3 Scaffolded Player

This assignment tasks the student to use a scaffolded song player program, where a given set of notes and their corresponding durations are read from a file and played sequentially. This assignment is an inspiration from Nifty's Melody assignment [28]. The assignment scaffold provides a pre-defined function responsible for playing a specific note for a specified duration of time. Students are tasked with implementing the functionality to parse the input file, and extract note-frequency pairs along with their durations.

The input file contains entries representing musical notes along with their respective durations, structured as follows: each line contains a note (e.g., F#3) followed by a duration (e.g., 200ms). The program reads this file, extracts note-duration pairs, and plays each note using the provided function for the specified

duration.

Unlike a full-frequency player, which is a later assignment in section 4.6, that encompasses the entire process of generating audio signals based on mathematical formulas (such as sine waves) and managing playback, this assignment focuses solely on reading note-duration pairs from a file and using a scaffolded function to handle adding the note to the `AudioClip` object. By completing this task, students reinforce their understanding of file IO operations and string manipulation.

#### 4.4 Sound Effects (Mixing, Fade, Volume Change, Clamping)

In this assignment, students implement basic audio effects such as mixing, fade, volume change, and clamping. For mixing, students would be tasked with combining two or more audio files. Arrays are used as `AudioClip` objects containing a list of channels and each channel contains a list of samples. For this assignment, the audio to be created will use one channel. To mix two samples you take the average (or a weighted sum) of the two values.

The fade effect could be introduced to teach students how to change the amplitude of audio samples over time. The goal is to fade one audio source out while another audio source fades in. Think about how song compilations will fade a song out near the end while fading in the next song so that the transition is smooth. As a sound is closer to the end, its amplitude will be dampened, and the next sound amplitude will be increased.

To teach volume change, students can apply a scalar value to the entire audio array, allowing them to understand how changing the amplitude uniformly affects the overall volume of the audio signal.

Lastly, the clamping effect could be implemented to teach students about limiting the amplitude values within a specific range. This is similar to the threshold assignment, but the values are limited to the threshold rather than being removed.

All assignments involving these effects include using conditional statements and array iteration to identify and modify samples.

#### 4.5 Visualize Volume Levels

Students will be tasked with visualizing the volume (amplitude) of an audio signal over time. The goal is to introduce the concept of representing time-varying data using a 2D array, where one axis represents time, and the other represents amplitude. Students could read in the audio file with a 1D representation of samples. Students then could create a *BRIDGES* `ColorGrid` object, which supports the creation of 2D images and has methods for coloring individual image cells.

For this image, the x-axis or columns correspond to time (time bins or frames), and the y-axis or rows represent the amplitude values at each time point. The idea is to separate the audio signal into frames to capture its change over time. Divide the audio signal into consecutive time frames, and for each frame, calculate the average amplitude level. Each frame/bin is represented by the current image column with the average calculated amplitude plotted on the y-axis.

Students can analyze the generated visualization to gain an understanding of how the volume of the audio signal changes over time. They can begin to recognize different parts of sounds and music that represent transitions from loud and soft segments. They could also recognize repetition and see which parts are similar to another.

#### 4.6 Full Frequency Player

Students can create a basic frequency player through the implementation of a sine wave generator that controls amplitude, frequency, and time of a note. This is an extension of the scaffolded player in section 4.3. Students are tasked with generating sample notes for specified durations that correspond to a sine wave, utilizing a given mathematical formula. The notes are also read from an external file describing notes and durations in the same way as the scaffolded player.

Students are also tasked to adjust the start and end parameters of the sine wave to correspond to specific time intervals measured in seconds, thus allowing for the generation of sine wave samples over a defined duration. This enables the creation of a sine wave audio clip spanning a specified time range playing multiple notes, effectively serving as the foundation for a basic frequency player. Students are encouraged to implement smooth transitions between successive samples enhancing the quality of the frequency player's output.

Overall, students creating a basic frequency player develop proficiency in several fundamental programming concepts. Firstly, it aims to increase understanding of modular programming by utilizing methods or functions to compartmentalize code, thereby enhancing code organization and reusability. The task enhances proficiency in data type manipulation and conversion, particularly through the casting between different data types.

#### 4.7 Instrument Types

To teach the use of objects and encapsulation, students can create basic musical instruments that have their attributes. Students create an instrument class, where they encapsulate attributes like pitch, amplitude, and waveform type. Each class is structured to represent a distinct musical instrument, introduc-

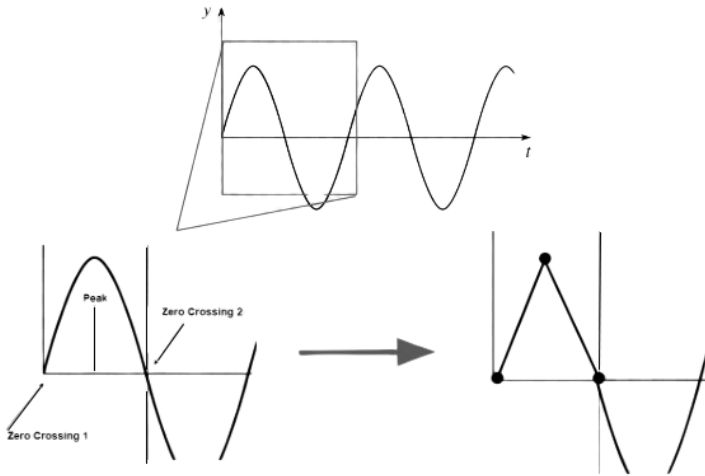


Figure 2: A sample illustration of compression one peak from a sin wave using two zero crossing points and a local peak. This creates a linear signal between the peaks and zero crossings.

ing students to the fundamental principles of object-oriented programming. Students could create an Instrument base class with individual instrument classes to teach polymorphism. Within these classes, students implement a play method that generates the instrument’s sound based on specified attributes, thereby delving into the concept of encapsulation and method definition within objects. Students proceed to instantiate objects from these classes, creating instances of various musical instruments and customizing their attributes. The assignment advances to the concatenation of these objects into a single AudioClip allowing for the playback of multiple instruments to create musical compositions through mixing and adding to different channels. Students would already know how to perform mixing from previous assignments.

#### 4.8 Basic Compression

This audio-processing assignment is focused on wavefront compression through zero crossings with activation thresholds,

Beginning with the loading and representation of audio data as a sequence of samples, students conduct a detailed analysis of the waveform, pinpointing instances of zero crossings where the audio waveform intersects the zero amplitude line, and marking significant signal transitions. The compression algorithm, designed by the students, captures the audio data by storing only the

temporal positions of zero crossings and the peak values between zero crossings (see Figure 2).

This approach aims to reduce the number of stored samples in the compressed representation while retaining perceptually relevant features. Students experiment with various activation threshold values, exploring the trade-offs between compression efficiency and the preservation of audio information. Decompressing the signal involves reconstructing the signal using the positions of the zero crossings, the peaks between the zero crossing, and using a linear function (line segment to approximate the original samples) removed during the compression phase. Decompression and playback assess signal preservation after compression.

Instructors can look at compression effectiveness, considering factors like compression ratio, file size reduction, and perceptual quality, when grading this work. Students gain a comprehensive understanding of the impact and trade-offs associated with lossy compression, and see if certain signals are worth the quality loss. Students could compete to see who can make the best audio compression algorithm to help enforce engagement with that assignment and other students. Instructors can also relate audio compression to image/video compression (such as JPEG, MPEG) in their coverage of this topic.

## 5 Conclusion

We have presented a sequence of audio-themed assignments appropriate for CS1/CS2 level courses supported as part of our *BRIDGES* toolkit, that allows for students to easily use and edit audio Wavefront files for content-based learning that promotes engagement. These assignments are also easy to incorporate in the course for instructors with an already completed repository of assignments along with the solutions and scaffolds needed. The assignments map to the most commonly covered topics and learning objectives within CS1, with flexibility to customize each assignment by student interests, and available for wider use in Java, Python, or C++. Each assignment builds on prior knowledge from the one before it with an end-of-class project incorporating all knowledge.

Some limitations of this work is that it has not been incorporated into a full CS1 course yet. The tool has been used with individual assignments by current and past *BRIDGES* users for multiple semesters, with positive feedback from both instructors and students. Also, currently, our tool only supports the generation of our own signals from scratch or by importing files in WAV formats. Extensions to other audio formats can be envisioned. In the future, we also plan on adding additional assignments with more variations.

## References

- [1] J. C. Adams, B. D. Allen, B. C. Fowler, M. C. Wissink, and J. J. Wright. The sounds of sorting algorithms: Sonification as a pedagogical tool. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, SIGCSE 2022, page 189–195, New York, NY, USA, 2022. Association for Computing Machinery.
- [2] J. D. Bayliss and S. Strout. Games as a flavor of cs1. In *ACM SIGCSE Bulletin*, volume 38, pages 500–504. ACM, 2006.
- [3] A. Beckman, M. McQuaigue, A. Goncharow, D. Burlinson, K. Subramanian, E. Saule, and J. Payton. Engaging Early Programming Students with Modern Assignments Using BRIDGES. volume 35, page 74–83, Evansville, IN, USA, apr 2020. Consortium for Computing Sciences in Colleges.
- [4] BRIDGES Development Team. BRIDGES Assignment Repository. <http://bridgesuncc.github.io/newassignments.html>, 2022.
- [5] BRIDGES Development Team. BRIDGES Website. <http://bridgesuncc.github.io>, 2022.
- [6] K. B. Bruce. Controversy on how to teach cs 1: a discussion on the sigcse-members mailing list. In *ACM SIGCSE Bulletin*, volume 36, pages 29–34. ACM, 2004.
- [7] S. Buchanan, B. Ochs, and J. J. LaViola Jr. Cstutor: a pen-based tutor for data structure visualization. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 565–570. ACM, 2012.
- [8] J. Burg, J. Romney, and E. Schwartz. Computer science "big ideas" play well in digital sound and music. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 663–668, New York, NY, USA, 2013. Association for Computing Machinery.
- [9] D. Burlinson, M. McQuaigue, A. Goncharow, K. Subramanian, E. Saule, J. Payton, and P. Goolkasian. Bridges: Real world data, assignments and visualizations to engage and motivate cs majors. *Education and Information Technologies*, 2023.
- [10] D. Burlinson, M. Mehedint, C. Grafer, K. Subramanian, J. Payton, P. Goolkasian, M. Youngblood, and R. Kosara. BRIDGES: A System to Enable Creation of Engaging Data Structures Assignments with Real-World Data and Visualizations. In *Proceedings of ACM SIGCSE 2016*, pages 18–23, 2016.



- [11] S. Dahlby Albright, T. H. Klinge, and S. A. Rebelsky. A functional approach to data science in cs1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 1035–1040. ACM, 2018.
- [12] A. Delman, A. Ishak, L. Goetz, M. Kunin, Y. Langsam, and T. Raphan. Development of a system for teaching cs1 in c/c++ with lego nxt robots. In *FECs*, pages 396–400, 2010.
- [13] P. Dewan. How a language-based gui generator can influence the teaching of object-oriented programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 69–74. ACM, 2012.
- [14] P. Drake and K. Sung. Teaching introductory programming with popular board games. In *Proceedings of ACM SIGCSE*, SIGCSE '11, pages 619–624, 2011.
- [15] J. Freeman, B. Magerko, D. Edwards, and L. Ikkache. Earsketch, a web-application to teach computer science through music (abstract only). pages 640–640, 03 2017.
- [16] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, and E. Fruchter. Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 85–90, New York, NY, USA, 2014. Association for Computing Machinery.
- [17] I. Greenberg, D. Kumar, and D. Xu. Creative coding and visual portfolios for cs1. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 247–252. ACM, 2012.
- [18] P. J. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 579–584. ACM, 2013.
- [19] M. Guzdial. A media computation course for non-majors. In *Proceedings of the ITICSE 2003*, pages 104–108, 2003.
- [20] O. A. Hall-Holt and K. R. Sanft. Statistics-infused introduction to computer science. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 138–143. ACM, 2015.
- [21] M. Hendelsman, W. Briggs, N. Sullivan, and A. Towler. A measure of college student course engagement. *Journal of Educational Research*, 98(3):166–175, 2005.

- [22] D. Horton, M. Craig, J. Campbell, P. Gries, and D. Zingaro. Comparing outcomes in inverted and traditional CS1. In *Proceedings of the ITICSE 2014*, pages 261–266, 2014.
- [23] Joint Taskforce on ACM Curricula. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science (Page 513)*. ACM/IEEE Computer Society, 2013.
- [24] A. N. Kumar. The effectiveness of visualization for learning expression evaluation. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 362–367. ACM, 2015.
- [25] W. M. Kunkle and R. B. Allen. The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1):3, 2016.
- [26] NCWIT, 2018. <https://www.engage-csedu.org/>.
- [27] N. R. Nguyen, H. Padhye, E. Stein, and G. Bull. Tunescope: Engaging novices to computational thinking through music. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, SIGCSE 2022*, page 1181, New York, NY, USA, 2022. Association for Computing Machinery.
- [28] A. Obourn and M. Stepp. Melody. <http://nifty.stanford.edu/2015/obourn-stepp-melody-maker/>.
- [29] N. Parlante. Nifty assignments, 2018.
- [30] J. Pirker, M. Riffnaller-Schiefer, and C. Gütl. Motivational active learning: Engaging university students in computer science education. In *Proceedings of ITICSE*, pages 297–302, 2014.
- [31] C. Reis and R. Cartwright. Taming a professional ide for the classroom. In *ACM SIGCSE Bulletin*, volume 36, pages 156–160. ACM, 2004.
- [32] S. Schaub. Teaching java with graphics in cs1. *ACM SIGCSE Bulletin*, 32(2):71–73, 2000.

# Mitigating Use of Artificial Intelligence in Student Assignments\*

Michael Jonas  
Applied Engineering and Science Department  
University of New Hampshire  
88 Commercial Street  
Manchester, NH 03101  
mcy59@unh.edu

## Abstract

With the proliferation of advanced Artificial Intelligent systems trained on robust language models, it is becoming more difficult to discern acts of plagiarism in the classroom. Though some of this falls under the academic misconduct policy of an institution, it can be confusing to students as to what qualifies as proper use. Students use tools like Grammarly, to improve their writing, and find more advanced AI tools, such as ChatGPT as an additional resource. For faculty to simply ban the use of these tools, creates an unworkable model. Embracing them can also be problematic as not all educational material benefits from a flipped approach. In this paper we discuss changing the dynamic by applying targeted assessments that incrementally address elements in an assignment, to affirm the work that students submitted and discourage use of improper tools that don't assist in learning.

## 1 Introduction

The problem of plagiarism is not new to student learning. It's not always a result of students trying to cut corners to avoid doing work. Sometimes students legitimately feel that collaborating with others creates a better learning

---

\*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.

environment for them to understand difficult material. With the advances in Artificial Intelligence (AI) systems, a means of collaboration can now take on the form of a single student, working on their own, getting inappropriate help. It differs from online sites offering either services to do work, or access to a trove of past assignments, in that the use of AI doesn't necessarily feel like a direct moral conflict. A student isn't really getting someone else to do their work, and they aren't paying for past assignments. They are simply querying an advanced search engine for help and use its results in the assignment's submission. A student confronted with the potential academic misconduct violation, often claims that they see little difference between using a grammar checking tool, such as Grammarly, and a well trained AI such as ChatGPT. The difference is more in how students query what they need from it. This can be especially true for international students, who attest that using these tools helps them refine answers that might otherwise sound disjoint with their lack of proficiency in the English language.

### **1.1 Academic Setting**

The University of New Hampshire at Manchester (UNH-M) is an urban commuter campus and one of seven colleges of the University of New Hampshire (UNH). It is apart from the main campus, located in Manchester, the largest city in the state, and has an enrollment of about 1,100 students. The college's mission is to provide a liberal arts education and integrates both practical and professional experience into the classroom. The college offers majors in social science, humanities, applied sciences, and engineering. The student population is made up of undergraduate students, primarily drawn within the state from traditional routes of high school graduates, and non-traditional routes of two-year institutions. More recent, with a focus on developing professional graduate programs, the college has seen growth in international students, which has added significant challenges to an already complex environment. A fundamental principle of the college is in the value of face-to-face instruction. Class focus is both in traditional learning as well as integrative lab experience where students learn to work together in finding solutions to problems. Collaboration is a key element in the curriculum and is fostered in the classroom. As a commuter campus, many classes meet once weekly, for 3 hours, so designing an engaging learning experience is critical in ensuring student success. It is important to allow students the ability to utilize resources that help foster this environment, so navigating online learning tools, including AI-based systems like ChatGPT, is an important balancing act.

## 2 Student Misconduct

Confronted with seemingly endless online informational resources that students have access to, how does one navigate this without devolving into constant angst about whether students are doing their own work and learning the material? A common approach is to develop a concise academic misconduct policy, apply appropriate punishment to various levels of offense, and hope that fear of consequences result in reducing the number of infractions. All too common, students conflate the issue with confusion, denying that their intent was to break the rules, and argue there was lack of clarity and guidance by the instructor. Additionally, the work to submit each case not only impacts faculty workload, but can also be emotionally draining when confronted with these type of issues. At UNH-M, this increasingly became a problem as the international student population grew in its graduate programs. As many international students came from the same countries, the homogeneity of the communities became problematic. Unwanted collaboration was rampant, as many students in the same cohort roomed together. Worse, second year students would also be available to share past work with their newer counterparts. More recently, the utilization of online AI tools compounded the problem. In a recent first-year cohort, more than half the students were flagged by a single academic misconduct infraction, with some accruing multiple strikes. The initial approach was to let students admit to the misconduct and write a reflection of what they did, so both faculty and student could have a more positive conversation about how to move forward. UNH recently revised its academic misconduct policy, giving flexibility in how an instructor dealt with infractions, allowing students to redo parts of assignments, rather than flag the misconduct. If a student chose not to admit they plagiarized, an official case was submitted, which required substantial paperwork to sufficiently report the case. Those that denied they did anything wrong reasoned that they used tools such as ChatGPT merely for improving language; that the main premise of the work was theirs. To compound this, as graduate programs saw rapid growth, some instructors dealing with these issues were relatively new and inexperienced adjunct faculty. These instructors had less interest in finding a solution to the cause; instead focusing more on removing students who were not doing their own work. With residential faculty also taking on the role of graduate advising, students confronted with failing a course required extra time to help navigate the impact their misconduct had on their academic standing. All of this caught faculty by surprise and negatively impacted the graduate programs at UNH-M. There were no easy solutions, as international students failing a course could not be expelled from it since their visa status required fulltime enrollment. How does a student continue in a course, knowing the rest of the semester will not impact their failing grade? This raised serious questions

about how to proceed with some of the graduate programs.

### 3 Approach

A literature search using the terms 'artificial intelligence' and 'plagiarism' yields many academic papers addressing methods and techniques of detecting instances of machine generated work. Foltýnek, Meuschke, and Gipp [2] surveyed available literature in the area of plagiarism detection in 2019, but their work does not contain references to AI-based online search engines like ChatGPT, showing how quickly that technology has manifested itself. It's an area that one feels like they are quickly falling behind in. This paper looks at a more preventative approach, expanding on the work of Hwang and Gibson [3] who developed an assessment framework using 20-minute quizzes to test students on their understanding of the material and to help reduce occurrences of academic misconduct, regardless of whether machine or human generated. They studied different ways to integrate these assessment quizzes into the overall student grade, and though their technique improved upon common models of grading, it still had drawbacks. Not only were their assessments too general, asking student to repeat parts of assignments, the grading scheme could also unfairly weight the entire assignment more towards the assessment quizzes, A more recent study by English [1] reinforced the need to ensure that assessments do not penalize students who struggle with the same material on both the assignment and the assessment. To address these shortcomings, assessments were designed to be targeted to a student's work. By refining Hwang and Gibson's idea into this more individualized model, the grading rubric is more closely integrated with the assignment so that it does not unduly penalize students twice. These assessments are also short 20-minute quizzes, with questions specific to the assignment, without necessarily having students repeat the work. More importantly, how these assessments impact student grades is different, and does not change the existing grading rubric for the assignment.

#### 3.1 Assessing Student Work

A question can be as simple as asking students in a coding class to describe the input format of a file they read in. It may ask what functions were developed to solve the problem. Or, it may require students to write part of an algorithm where the solution of their assignment was reliant on a specific one. If a student was unable to solve the assignment fully, assessment questions covering the parts that were not completed could be skipped. Not only can points be deducted when a student fails to properly answer an assessment question, points can also be earned back if a student answers a similar question on the assessment that they failed to answer on the assignment. This reinforces a more

positive model of assessments than just a means to catch academic misconduct. Assessments can be viewed as incremental in nature, only targeting small subsets of a problem. Their goal is not to draw inferences from an assignment, as that is left to general examination. So assessments can be seen as low stakes quizzes, and they do not have any point values associated with them, utilizing the assignment rubric as guidance to what values will be assessed. There isn't a one-to-one correlation, necessarily, mapped from assessment question to assignment. This is because some assessment questions may focus on an assignment element that is worth very little, but failure to answer may lead to an overall negative assessment over the entire assignment. For instance, if 5% of an assignment asks students to solve numeric floating point precision in a C++ coding problem, the assessment may ask what specifically they did to solve it. If a student who successfully implemented this floating point precision is unable to answer that question, it creates doubt about the authenticity of the entire assignment. To solve it, a student would have researched a specific library, included it, and then used a specific set of directives to accomplish the task. This would not have been covered in the classroom and the expectation is that students would have researched this on their own and added it in to achieve the correct output, or ignore it and accept the small point loss. A student solving this would know exactly what library they found and included. If their assessment answer either claimed that they didn't solve it, or discussed something completely unrelated, it would yield a preliminary zero for the entire assignment and a meeting with the instructor. An important component in the assessment model is for students to have the opportunity to discuss any misalignments between their assignment and what answers they gave on their assessment. A student would get to demonstrate, in a face-to-face meeting, that they knew what they had done in the assignment. This is not meant as a negative component, but rather gives the ability for a student, who feels confident that they knew an element that they failed to properly address on an assessment, to clarify so they could earn any lost points back. Especially in cases where assessment answers cause a significant point-loss on their overall grade, a student is encouraged to discuss the difference with the instructor.

### 3.2 Integration

An initial run of this approach had already been in place for several years, before tools like ChatGPT became prevalently known. A graduate level Information Security course had seen high cases of plagiarism, primarily because it was offered at least twice, sometimes three times a semester, including during the summer term, resulting in a large trove of past assignments being shared by students [5]. It was difficult to change all assignments on a per-semester basis, especially since a cryptographic sandbox was developed where students

had to modify parts of it for their learning goals [4]. Though the course was cross-listed with an undergraduate equivalent, only graduate students were assessed on their work. In 2023, two more courses were included in the trial and undergraduate students in the Information Security class were also asked to do the assessments. The other two courses were a standard Data Structures course and a hybrid Machine Architecture and Systems course. The latter spends the first half of the semester below the CPU learning about machine code, and the second half right above it learning about the basics of operating systems. Both new courses have a robust coding component, with Data Structures using C++, and Machine Architecture and Systems covering a variety of topics from machine code programming, to operating system level scripting with languages like bash and Perl.

|                                | Problem Questions  | Assessment Questions   | Comments   |
|--------------------------------|--|--|--|
| Information Security           | <ul style="list-style-type: none"> <li>Pick three websites you use and discuss their password policies.</li> </ul>   | <ul style="list-style-type: none"> <li>Which website password policy was most secure and why?</li> </ul>   | <ul style="list-style-type: none"> <li>These types of question see how well students know their own work.</li> </ul>   |
|                                | <ul style="list-style-type: none"> <li>Develop a simple cryptographic hash, based on a modified checksum.</li> </ul>   | <ul style="list-style-type: none"> <li>Write JavaScript function that computes the ASCII product of a string.</li> </ul>   | <ul style="list-style-type: none"> <li>If students submitted working program, did they code it.</li> </ul>   |
| Data Structures                | <ul style="list-style-type: none"> <li>Create a searchable phonebook with arrays, using bubble sort and binary search to find items quickly.</li> </ul>          | <ul style="list-style-type: none"> <li>What was in header of text-based input file?</li> <li>Complete partially written binary search.</li> <li>Write bubble sort's swap.</li> </ul> | <ul style="list-style-type: none"> <li>Students who did work easily know this.</li> <li>Had to compute mid and left/right values.</li> <li>Easy if they wrote sort.</li> </ul>                 |
|                                | <ul style="list-style-type: none"> <li>A more advanced phonebook that used a simple linked list to insert items in sorted order, and search linearly.</li> </ul> | <ul style="list-style-type: none"> <li>How many functions in your main class?</li> <li>Name instance variables in the Link class.</li> <li>Write Link traverse code.</li> </ul>      | <ul style="list-style-type: none"> <li>Asked occasionally to verify they wrote it.</li> <li>This is also <u>similar</u> to the first question.</li> <li>A <u>simple</u> while loop.</li> </ul> |
| Machine Architecture & Systems | <ul style="list-style-type: none"> <li>Do binary arithmetic.</li> </ul>  | <ul style="list-style-type: none"> <li>Repeat, but simpler form.</li> </ul>  | <ul style="list-style-type: none"> <li>Straight forward.</li> </ul>  |
|                                | <ul style="list-style-type: none"> <li>Given hex dump, disassemble and <u>run machine code program</u></li> </ul>  | <ul style="list-style-type: none"> <li>Disassemble a hex dump.</li> <li>Run a short machine code example.</li> </ul>   | <ul style="list-style-type: none"> <li>The assessments would ask simpler forms of assignment questions.</li> </ul>   |
|                                | <ul style="list-style-type: none"> <li>Write a Perl program to hash names to numbers.</li> </ul>   | <ul style="list-style-type: none"> <li>Write Perl program that prints content of text file.</li> </ul>   | <ul style="list-style-type: none"> <li>Asks for simpler element of assignment.</li> </ul>  |

Table 1: Sample assessment questions

The Information Security course had already demonstrated that this type of assessment helped reduce instances of academic misconduct. The assessments consisted of quizzes for each problem set that students submitted. Problem sets were divided into discussion questions that could be researched through online searches, and problem solving questions that required students to add small amounts of code. This yielded two types of questions on assessments: ones that were specific to each student's answer, and ones that had students



work on small coding elements (see examples in table 1). It was expected that all students would answer assessments questions on areas from the discussion part, but could skip the problem solving ones if they were unable to do so on the assignment. Students knew they would not be double penalized for things they didn't know, which alleviated any stress associated with the assessment mechanism. The task of building assessments was more challenging in pure coding courses, as thought had to be given on how to design questions that students could do within the 20-minute time allotment but still be in-depth enough to cover the entirety of the assignment. As stated previously, a mix of questions were devised, some ensuring students understood the premise and were familiar with the framework of the assignment, and some getting to the core component by asking to re-code something related, though be sufficiently different so students weren't just memorizing code they submitted. Table 1 lists different examples of assessment questions used in the two undergraduate courses.

## 4 Observations

After five years of using the targeted assessment model in the graduate version of Information Security, a good sample size, it was clear that the technique helped reduce instance of plagiarism. Over that five year period, the method has been refined and is now more targeted than when originally introduced. Violations became rare and when they did occur, there was no need to submit an academic misconduct case, as most cases were isolated, and limited. In one instance, two students submitted the same coding solution to a cryptographic hash function, with the usual attempt to hide their collaboration by changing variable names. Both students could not repeat a simpler form of that code on the assessment, and therefore did not receive any points on that part of their submission. As this drew attention to the instructor, it was easy to follow future submissions by those students, and no further collaboration occurred. One student started skipping the coding parts, while the other continued to struggle on theirs. Since the grading rubric was designed so that students wouldn't necessarily fail the course if they struggled on the coding part, it underscored that unwanted collaboration, with another students, or an online system, was neither needed nor beneficial. The use of assessments also helped change the culture of the course, as students stopped finding ways to circumvent the academic misconduct policies. In past years, due to the homogenous environment that many international graduate students found themselves in, passing old assignments around was a major issue. More recently, as other computing courses at UNH-M experienced an increase in ChatGPT-based infractions, none arose in the course. Only a handful of misconduct cases have

been submitted for a course that runs two to three times each year (i.e. about 12 semesters in total). A similar experience has been observed in the two undergraduate courses added in 2023. Though the number of academic misconduct cases in previous semesters was minimal compared to the graduate course, there were still occurrences that were caught and reported. Additionally, the possibility of AI-assisted submissions was something of concern, since they might be harder to track down. There have yet to be any infractions, save for one student turning in a coding lab in the Machine Architecture & Systems course that they could not repeat on the assessment. That student lost full credit for that part of the assignment, with the option to regain those points if they could show, in a live session, that they had the ability to create such a complex solution. The student chose not to follow up and did not repeat any further offenses. There were also a significant number of cases where students answered questions on the assessments, that they missed on the assignments, thus regaining some lost points. This helped turn the assessment mechanism into a more positive experience for students, allowing for a second opportunity to demonstrated what they knew.

## 5 Conclusions

This paper describes a model of targeted assessments to address plagiarism, especially from harder to detect online AI-based systems. This work is presented as an experience paper, with observations from a broader trial run of three computing courses. By forcing students to demonstrate what they learned, it reduces the benefit of taking shortcuts. Preliminary observations demonstrate that this technique works, as occurrences of academic misconduct cases over the five-year run of the Information Security course have been reduced to rare instances. With the two newly added undergraduate courses showing similar student behavior, it suggests that this model of assessing student work can be a tool that helps an instructor better handle the challenges of the modern education ecosystem. Though designing assessments is dependent on various criteria, for this work, it has been applied to courses with a heavy coding element as well as courses with a mix of problem solving and research elements alongside a lighter focus on coding. An important factor is scheduling class time for the assessments. With UNH-M being a commuter campus, and many classes running for 3 hours once a week, adding a 20-minute assessment does not negatively impact student class experience. With a more traditional model of 50-minute sessions, three times a week, a slightly shorter assessment duration might be a better fit. The time needed for assessments depends on how many questions are asked, and can be adjusted accordingly. Another important element is how to share assessment scores with students. The author chose

not to return the assessment quizzes themselves, as that reduces the ability of students to post them online and build up a database, something that could circumvent their impact. Instead, students are given written feedback on their assignment whenever there is a mismatch between what they submitted and what they answered on the assessment.

## References

- [1] Rosanne English. “Designing computer security assessments to reduce plagiarism”. In: *Proceedings of the 3rd Conference on Computing Education Practice*. 2019, pp. 1–4.
- [2] Tomáš Foltýnek, Norman Meuschke, and Bela Gipp. “Academic plagiarism detection: a systematic literature review”. In: *ACM Computing Surveys (CSUR)* 52.6 (2019), pp. 1–42.
- [3] C Jinshong Hwang and Darryl E Gibson. “Using an effective grading method for preventing plagiarism of programming assignments”. In: *ACM SIGCSE Bulletin* 14.1 (1982), pp. 50–59.
- [4] Michael Jonas. “A cryptographic sandbox for an introductory security course in information technology”. In: *Journal of Computing Sciences in Colleges* 29.6 (2014), pp. 74–80.
- [5] Michael Jonas. “Addressing Plagiarism: How to Reduce Occurrences in a Computing Curriculum Through Targeted Assessment”. In: *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE). 2023, pp. 1275–1279.

# GitKit: Teaching Git and GitHub/GitLab Workflow in an Authentic Context\*

Workshop

Grant Braught<sup>1</sup>, Stoney Jackson<sup>2</sup>, Karl R. Wurst<sup>3</sup>

<sup>1</sup>Computer Science

Dickinson College

Carlisle, PA 12013

[braught@dickinson.edu](mailto:braught@dickinson.edu)

<sup>2</sup>Computer Science and Information Technology

Western New England University

Springfield, MA 01119

[hjackson@wne.edu](mailto:hjackson@wne.edu)

<sup>3</sup>Computer Science

Worcester State University

Worcester, MA 01602

[kwurst@worcester.edu](mailto:kwurst@worcester.edu)

## Abstract

The GitKit facilitates teaching Git and GitHub workflow in the context of an authentic Free and Open Source Software (FOSS) project. It is appropriate for use in software development courses ranging from high school through college. The GitKit is a snapshot of a FOSS project's artifacts (codebase(s), issues, etc.) packaged with student learning activities, an instructor guide, and a containerized development environment. The GitKit can be used to provide students with a first exposure to Git/GitHub in a few class sessions, or a more comprehensive experience over 4-6 sessions. The aim of this workshop is to familiarize faculty with classroom use of the GitKit and thus participating faculty should have prior familiarity with Git/GitHub. The majority of the workshop time will be dedicated to hands-on experience with the GitKit from both the student and instructor perspectives.

---

\*Copyright is held by the author/owner.

## **Presenter Biographies**

Grant Braught, Stoney Jackson, and Karl R. Wurst have integrated Humanitarian FOSS into portions of the computer science curriculum. They are the primary developers of the GitKit activities and associated technologies, and all three have used the GitKit in their classes (as have faculty at other institutions). They are continuing to develop other learning activities using the Kit philosophy and technology. They all maintain humanitarian, free and open source projects, including FarmData2 and LibreFoodPantry.

## **Intended Audience**

This workshop builds on the tutorial that was presented at CCSCNE 2023, by providing a more in-depth, hands-on, instructor-oriented experience. This workshop is intended for CS educators who wish to teach Git and GitHub workflow in the context of an authentic FOSS project. The GitKit provides an introduction to Git and FOSS contribution workflow that is appropriate for software development courses ranging from high school through college software engineering.

## **Materials Provided**

Attendees will be provided a link to a site that contains the GitKit, including all workshop materials, an instructor guide, classroom activities and student assignments. Information about other kits, ways to become involved, and contact information for the developers will also be provided.

## **Acknowledgements**

This work was supported under National Science Foundation Grants DUE-1225738, 1225688, 1225708, 2012966, 2013069, 2012979, 2012999, and 2012990. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

# Guiding Students to Discover CS Concepts and Develop Process Skills Using POGIL\*

Conference Workshop

Olga Glebova<sup>1</sup>, Ali Al-Farris<sup>2</sup> and Nanette Vielleux<sup>3</sup>

<sup>1</sup>School of Computing  
University of Connecticut  
Storrs, CT 06268

`olga.glebova@uconn.edu`

<sup>2</sup>Computer Science Department  
Worcester State University  
Worcester, MA 01602

`aalfaris@wrochester.edu`

<sup>3</sup>Mathematics, Computing and Statistics Department  
Simmons University  
Boston, MA 02115

`vielleux@simmons.edu`

There has been growing interest across CS education for evidence-based instructional strategies that improve learning and retention. Process Oriented Guided Inquiry Learning (POGIL) is one such strategy with a proven history across STEM disciplines. The CS POGIL community has grown rapidly over the past decade thanks to NSF support of multiple grants of educational research in Computer Science using this strategy; hundreds of CS teachers and professors have attended POGIL workshops to learn more about using and developing POGIL materials. A survey of CS POGIL practitioners found that instructors believe their students learn more, are more engaged and active, and develop better communication skills in POGIL classrooms [1]. The survey also found that obstacles include limited time, a lack of relevant materials, pressure to cover content, difficulty adapting teaching style, and resistance from students. Workshops like this one can help practitioners to answer these ques-

---

\*Copyright is held by the author/owner.

tions and to adopt POGIL more effectively and efficiently. This workshop is based on materials developed and used extensively by The POGIL Project (<http://pogil.org>).

## References

- [1] Helen H Hu et al. “Results from a survey of faculty adoption of process oriented guided inquiry learning (POGIL) in computer science”. In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 2016, pp. 186–191.

# Bridging the Empathy Gap: Hands-On Exploration of **H**uman-**E**mpathy **A**ccessibility **L**earning (HEAL) Interventions in Computer Science Education\*

Conference Workshop

Maggie Frechette<sup>1</sup>, Annabel Fisher<sup>1</sup>, Mohammad Shaikh<sup>1</sup>,  
Jon LeFrois<sup>1</sup>, Dmitry Burshteyn<sup>1</sup>, Sam Malachowsky<sup>2</sup>,  
Daniel Krutz<sup>2</sup>, Jami Cotler<sup>1</sup>

<sup>1</sup>Siena College

{ms28frec,ab24fish,ms18shai,jp29lefr,dburshteyn,jcotler}@siena.edu

<sup>2</sup>RIT

{dxkvse,samvse}@rit.edu

This workshop addresses the critical need for empathy development in computer science education, emphasizing the importance of understanding user needs, effective communication, collaboration, and inclusive design [9]. As 41 states across the United States have adopted accessibility policies, affecting 82% of the nation, the pursuit of inclusivity goes beyond mere compliance with regulations. However, despite the significant presence of disabilities, computer science students often remain minimally exposed to the challenges of developing technology through the lens of individuals with disabilities [5]. This deficiency highlights the critical importance of embedding empathy and user-centered perspectives into their educational framework. To address this deficiency, the workshop provides participants with hands-on experience through in-person accessibility interventions, showcasing the impact of Human-Empathy Accessibility Learning (HEAL) on undergraduate computing students. Supported by foundational research, these in-person interventions are shown to enhance empathy, understanding of technological barriers faced by individuals with disabilities, and foster a wide range of skills including technical proficiency, soft

---

\*Copyright is held by the author/owner.



skills, and empathy-related abilities crucial for career progression [2]. The workshop is structured as an immersive exploration of three distinct accessibility interventions, coupled with discussions on the application of validated scales for evaluating the interventions' impacts. A fundamental goal is for attendees to be capable of conducting these interventions in their classrooms, thus extending empathy among computing students in collegiate settings and equipping future computing professionals with the empathy, understanding, and inclusivity needed to navigate the challenges of the modern workplace [2, 5, 9].

## 1 Introduction

In the dynamic world of computer science education, empathy integration is becoming increasingly crucial [9]. Empathy plays a vital role in effective problem-solving and user-centered design and is crucial for understanding user needs and fostering inclusive design practices [4]. Our HEAL workshop, recognizing the need for understanding user needs and inclusive design, aims to fill the educational gap in addressing disabilities in technology development. It offers hands-on accessibility intervention experiences to cultivate empathy in undergraduates [2].

### 1.1 Workshop Structure

Our workshop, aimed at fostering a more empathetic and inclusive computer science education, begins with PLAN-O (presence, listening, attention, non-judgementalism – leads to openness) mindful introduction[1] to highlight the importance of empathy. Human-Empathy Accessibility Learning (HEAL) project's principles will provide a scientific overview of its rationale. The core is a 1.5-hour hands-on session which highlights 3 accessibility interventions, connecting participants to empathy and inclusive design. A 15-minute discussion follows on evaluation metrics culminating in key takeaways and potential HEAL applications in various educational contexts.

## 2 Workshop Session Agenda

- **Introduction and HEAL Project Overview (15 minutes)** The introduction will cover the importance of empathy in computer science education and a discussion of the current state of accessibility policies in the United States.
- **Hands-On Experience with Accessibility Interventions (1.5 hours)** Participants will be invited to explore three immersive accessibility inter-

ventions. The interventions include a client reveal, immersive hands-on empathy lab and RIT’s Accessibility Learning Lab (ALL) [3, 6, 7, 8].

- **Discussion: Psychometric Instruments (15 minutes)** An overview of the how the validated Perth Empathy Scale (PES), TEIQue 2.0 Emotional Intelligence, and the work motivation surveys are utilized to evaluate the impact of the interventions will be offered.
- **Q&A, Group Discussion (15 minutes)** Participants will be given an opportunity to share experience and insights and invited to a collaborative conversation about integrating empathy into computer science education.
- **Closing remarks and future implementations (15 minutes)** Key takeaways from the workshop along with a discussion on potential implementations of HEAL interventions in diverse educational settings.

**Requirements:** Participants are asked to bring a laptop for the hands-on experience.

**Target Audience:** Educators, curriculum developers, and administrators in computer science education interested in fostering empathy and inclusivity in their programs.

By the workshop’s conclusion participants will have gained practical insights into implementing HEAL interventions and evaluating their impact, contributing to a more empathic and inclusive computer science education landscape.

## References

- [1] Jami Cotler and Dmitry Burshteyn. *Experimental Methods. PLAN-O (presence, listening, attention, non-judgementalism – leads to openness) mindful introductions*, Class Lecture, Experimental Methods I, Siena College, Loudonville, NY, November 2023.
- [2] Jami Cotler et al. “Empathy-Driven Student Transformations: Bridging the Gap in Software Development for Inclusive User Experiences”. In: *Proceedings of Information Systems & Computing Academic Professionals (ISCAP)* (2023). Under review for Information Systems Education Journal.
- [3] Saad Khan et al. “Experiential learning in undergraduate accessibility education: Instructor observations”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 54–68.

- [4] Merlijn Koupprie and Froukje Sleeswijk Visser. “A framework for empathy in design: stepping into and out of the user’s life”. In: *Journal of Engineering Design* 20.5 (2009), pp. 437–448.
- [5] Jonathan Lazar, Daniel F Goldstein, and Anne Taylor. *Ensuring digital accessibility through process and policy*. Morgan kaufmann, 2015.
- [6] Yang Liu et al. “Do Users Act Equitably? Understanding User Bias Through a Large In-person Study”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE. 2023, pp. 83–95.
- [7] Heather Moses et al. “Experiential Educational Accessibility Modules”. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–3.
- [8] Weishi Shi et al. “Experiential learning in computing accessibility education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. 2020, pp. 250–251.
- [9] Anne Wallisch, JUAN CARLOS Briede-Westermeyer, and Marianela Luzardo-Briceno. “Fostering user-empathy skills of engineering students by collaborative teaching”. In: *International Journal of Engineering Education* 37.1 (2021), pp. 223–243.

## Presenter Biographies

**Maggie Frechette** is a 4th year undergraduate student studying computer science at Siena College and the team lead for this research at Siena College.

**Annabel Fisher** is a 3rd year undergraduate student studying psychology at Siena College. She has contributed to the team’s understanding of empathy and how to design the interventions to best build this skill.

**Jon LeFrois** is a 3rd year undergraduate student studying actuarial science and computer science at Siena College. He has contributed to the design of the study and has helped conduct this research.

**Mohammad Shaikh** is a 4th year undergraduate student studying computer science at Siena College. He has contributed to the design of the study and has helped conduct this research.

**Daniel Krutz** is the PI of the NSF-funded projects (#1825023, #2111152 and #2145010) that are devoted to creating the Accessibility Learning Labs

(ALL). Krutz has taught approximately ten different graduate and undergraduate software engineering courses and is the author of over twelve pedagogical research papers.

**Samuel Malachowsky** is the Co-PI of the project that is developing the described ALL accessibility labs. Malachowsky is a Sr. Lecturer at RIT and has authored 9 pedagogically focused publications. He holds a Project Management (PMP) certification and has authored a textbook on team leadership.

**Dmitry Burshteyn** is an Associate Professor of Psychology at Siena College, specializing in the psychology of decision-making and problem-solving, with a strong focus on accessibility training. His academic foundation is rooted in experimental psychology, where he has developed an expertise in research methodologies and statistical analysis.

**Jami Cotler** is an Associate Professor of Computer Science at Siena College with a research and teaching focus in career readiness, accessibility advocacy, empathy development for software engineers, and UX Design. She has over 25 years of experience in academia and industry with sixteen pedagogically focused publications.

# Designing Technical Presentation with Good Narrative Structure\*

Workshop

Paul Kim

Computer Science

Bridgewater State University, Bridgewater, MA 02325

p2kim@bridgew.edu

When computer science students graduate and work in industry, they need to communicate with others while working on projects. And one of the soft skills that is important in industry is conveying their ideas to others. But when students have a chance to talk about their ideas such as giving presentations in a class or at a public symposium, their presentations usually lack a good storyline (narrative) that delivers their points clearly to audiences. Therefore, we created a lecture targeting computer science students that teaches how to design an effective presentation, especially a technical presentation, with a clear narrative. In this workshop, we will introduce how we designed the lecture, and participants will be involved in a group activity that students did in the lecture to see the effectiveness of our teaching method.

## 1 ABT Framework

When students give technical presentations, it is important to show clearly why they did the projects because it makes audiences understand the importance of it and get more interested in their projects. But it is challenging for students to design such a storyline for their presentations because there is no specific guideline to follow. Therefore, in our lecture, we set up a practical guide that helps students build an effective narrative structure for their presentations using the framework called the ABT framework [1][2].

The ABT framework uses three words AND, BUT, and THEREFORE, and those words work as tools to build a clear narrative that can make presentation

---

\*Copyright is held by the author/owner.

effective. Simply saying, AND introduces the background of the project, BUT shows what kind of problem is in the project, and THEREFORE explains how to solve the problem in the project. By showing the problem in the BUT part of the framework, students become able to create a storyline that shows why this project is important effectively.

## 2 Workshop Description

This workshop will introduce the ABT framework and how to apply it to design a narrative structure for technical presentations. Then, participants will be involved in a group activity in which each group will choose one random computer science topic and design a presentation with the ABT framework. The activity will follow the schedule below:

1. Making groups
2. Choosing topics for the presentation
3. Designing BUT part of ABT framework
4. Designing AND part of ABT framework
5. Designing THEREFORE part of ABT framework
6. Building a whole narrative for presentation
7. Creating presentation materials with complete narrative
8. Giving a presentation to audiences

Via this workshop, we expect that participants will learn the effectiveness of the ABT framework in designing a technical presentation. Also, via the group activity, they will see from a student’s view about how to build a narrative with the framework so that they can help students more effectively in the future. All worksheets and other necessary resources will be provided during the workshop. Participants are required to bring their laptops.

## 3 Short Biography of Presenter

Dr. Paul Kim is an assistant professor of the computer science department at Bridgewater State University. His research interest lies in computer science education specifically about how to design thinking classrooms for computer science students. He is also interested in advising students to create games for advanced fields such as education and health care.

## References

[1] Randy Olson. Houston, we have a narrative: why science needs story. *The University of Chicago Press*, 2015.

[2] Randy Olson. The narrative gym: Introducing the abt framework for messaging and communication. *Independently Published*, 2020.

# Fundamental Cybersecurity Hands-on Exercises using an in-house Cyber Range\*

## Workshop

Giuseppe Sena, Ryan Fried, and Shamsi Moussavi  
Computer Science Department  
Massachusetts Bay Community College  
{gsena,rfried,smoussavi}@massbay.edu

This workshop offers participants hands-on Cybersecurity exercises conducted on MassBay Community College's (MBCC) in-house Cyber Range, made possible through NSF DUE: 2100114 grant funding, AttrACTing the Next Generation Cybersecurity Workforce (ACT). The exercises developed for this environment are utilized by community college students, who, in turn, assist high school students at our workshops as near-peer tutors and other MassBay students majoring in IT and Cybersecurity. Participants will gain insights into the creation of this Cyber Range, the network topology, and engage in Cybersecurity exercises in a fully-contained environment. Workshop participants should bring their own laptop (Windows or MAC). The maximum capacity for the workshop is 16 participants. The workshop will consist of:

**A little about ACT** The focus of the MBCC ACT project is to address the shortage of cybersecurity talent in Massachusetts. The primary goal is to recruit, retain, and graduate proficient students who enter the workforce as skilled employees, with a special focus on underrepresented students, particularly women. Collaboration with regional high schools, industry leaders, and security experts is emphasized. The ACT ecosystem enables students to participate in Cybersecurity competitions, internships, hackathons, and other activities aimed at mastering the necessary knowledge and skills for Cybersecurity jobs. Through the ACT project, we have strengthened our partnership with regional industries by: developing labs with their products for our courses, having employees mentor our students, providing scholarships and internships, and collaborating with us on special events such as presentations, panels, and festivals.

---

\*Copyright is held by the author/owner.

**The Cyber Range (CR)** A Cyber Range is a controlled, interactive infrastructure providing students with virtual environments for realistic, hands-on cybersecurity labs and exercises. Establishing a CR environment at MBCC, with support from University of Massachusetts Lowell Cybersecurity faculty, provides 20 students with the opportunity to practice activities such as infrastructure defense, vulnerability assessment, and access control in a safe, isolated environment.

**Hands-on exercises**

a. **A Tour the of MBCC’s CR**

Provide a high-level overview of the CR, composed of two (2) physical servers running an open-source Type 1 Hypervisor (**XCP-ng**). Describe the logical & physical topology of the range, server configuration, services provided, and other technical details. Provide information regarding the testbed network developed in conjunction with the CR.

b. **Connecting Remotely to MBCC’s CR**

Participants will learn about:

- the purpose of a Cyber Range, grasp basic networking concepts crucial for a cybersecurity career, and gain insight into the lab topology and IT infrastructure of the MassBay Cyber Range.
- remotely connecting to the CR from their own laptops (Windows & MAC) to use several VMs (Windows & Linux) on the CR servers.
- IP addresses, employing the **ping** command to check machine status, and utilizing an IP Scanner to discover other machines on the network.
- Virtual Machines (VMs), particularly **Kali Linux**, a Linux distribution tailored for penetration testing. They’ll learn how to use Kali VMs for lab exercises, access vulnerable VMs on the private network, and connect to assigned Kali VMs using terminal server protocols like SSH and VNC.
- basic Linux commands and operations, such as navigating directories, creating and deleting files and directories, and using sudo for administrative tasks.

c. **Network Reconnaissance on the Testbed Network**

Participants will use open-source reconnaissance tools (like NMAP) to scan the network and specific servers to identify running applications. They will research vulnerabilities for specific versions of applications to conduct experiments.



d. **Capturing Passwords Hashes**

Participants will learn about commonly used penetration testing tools and how to capture and crack password hashes on your network. Identify best practices to defend your network against these types of attacks.

e. **Analyze Logs as Blue Teamer**

Participants will read applicable logs on the victim machine to reconstruct the attack.

**Complete survey**

**Giuseppe Sena**

Professor of Computer Science and Cybersecurity,  
Co-PI on ACT project, Computer Science Department Chair, and Instructor/-  
Manager of the MBCC's Cisco Networking Academy.

**Ryan Fried**

Adjunct Professor at MassBay, and Senior Security Consultant at Google  
Senior Personnel on ACT project, and Senior Security Analyst, specializing in  
security automation, network segmentation and purple teaming.

**Shamsi Moussavi**

Professor of Computer Science and Cybersecurity,  
PI for ACT project, Director of the Center for Cybersecurity Education at  
MassBay.

# Navigating Feasibility: Choosing Service-Learning Projects for Academic Fit\*

Conference Tutorial

Stan Kurkovsky<sup>1</sup>, Chad Williams<sup>1</sup>,  
Mikey Goldweber<sup>2</sup>, Nathan Sommer<sup>3</sup>

<sup>1</sup>Central Connecticut State University, New Britain, CT 06050

{kurkovsky,cwilliams}@ccsu.edu

<sup>2</sup>Denison College, Granville, OH 43023

mikeyg@denison.edu

<sup>3</sup>Xavier University, Cincinnati, OH 45207

sommern1@xavier.edu

Service learning can be viewed as a strategy in teaching and learning where meaningful community service is integrated with instruction and reflection. Service learning experiences can go a long way to enrich student learning, expose them to the concept of civic responsibility, strengthen communities, and show them the role that their profession can play in contributing to the common good [3]. Participation in service learning projects has been shown to strengthen students' interest in computing and related careers, especially among female and minority students [1].

Instructors and institutions looking to adopt service-learning projects face a broad range of questions. How to embed a project into one's existing curriculum without adding a new course? How best to supervise students, provide meaningful formative feedback, and assess and evaluate their work? How to work with the project partner to assess the project scope and ensure its feasibility? How to evaluate project success and make sure the project partner gets what they need? How to provide project maintenance after delivery?

Many existing practices of working with externally sourced projects often rely on established frameworks rooted in and supported by existing institutional infrastructures, additional staff, and technical resources. However, many institutions may not have such resources and/or experience. Those looking to

---

\*Copyright is held by the author/owner.

adopt such practices can often find reports about positive experiences with externally sourced projects, e.g. [2], but none of them provide the depth of detail that would enable an institution to readily adopt that work.

We are currently working on a larger project aimed to equip instructors with curricular materials, how-to guides, sample project portfolios, and ample engagement and training opportunities to adopt a robust service-learning framework with minimal resources. A successful service-learning experience depends as much on following a curricular process, as it does on a careful choice of a project partner, right project scope, fit of student capabilities, and many other characteristics.

This tutorial will introduce our service-learning project feasibility assessment model. Participants will be guided through a set of structured questions designed to facilitate meaningful discussions with project partners. These discussions aim to collaboratively complete a scoring rubric, providing an objective evaluation of the suitability of service-learning projects. The tutorial's focus is on aiding instructors to make informed decisions regarding course fit and identifying the most suitable projects for their institution and program curriculum. We will present a hands-on case study where participants review one or more potential projects with a hypothetical external partner and determine the fitness of the project within their academic program.

The tutorial builds on the experience accumulated by CCSU's Software Engineering Studio which connects community project partners with teams of 4-5 seniors working on software development projects spanning one or several semesters. Since 2014, the Studio facilitated over 65 distinct projects and engaged over 500 students.

## Acknowledgements

This work is supported in part by the National Science Foundation awards 2315322 and 2315323.

## References

- [1] Jill Denner. What predicts middle school girls' interest in computing? *International Journal of Gender, Science and Technology*, 3(1), 2011.
- [2] Nicole Herbert. Reflections on 17 years of ict capstone project coordination: Effective strategies for managing clients, teams and assessment. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18*, page 215–220, New York, NY, USA, 2018. Association for Computing Machinery.
- [3] Sarena Seifer and Kara Connors. *Faculty Toolkit for Service-Learning in Higher Education*. Learn and Serve America's National Service-Learning Clearinghouse, 2007. <https://community-wealth.org/content/faculty-toolkit-service-learning-higher-education>.

# Incorporating Computing for Social Good in Education

Conference Tutorial

Heidi J. Ellis<sup>1</sup> and Gregory W. Hislop<sup>2</sup>

<sup>1</sup>Western New England University  
Springfield, MA

`ellis@wne.edu`

<sup>2</sup>Drexel University  
Philadelphia, PA

`hislop@drexel.edu`

The ACM/IEEE/AAAI CS Curriculum Recommendation, CS2023, outlines a computer science curriculum for undergraduates. As a supplement to CS2023, a series of peer-reviewed articles has been collected to provide guidelines for computer science educators by addressing various aspects of the design and delivery of the curriculum. The Computing for Social Good in Education (CSG-Ed) article outlines a variety of ways that computing instructors can incorporate CSG into education. These approaches range from modifying a single assignment to implementing CSG throughout a curriculum.

This tutorial will provide computing instructors of varied experience levels with a variety of approaches to incorporating CSG in their classrooms. The tutorial will provide an overview of the importance of CSG-Ed and its role in CS2023, a discussion of approaches for including CSG in the classroom, examples of CSG-Ed and a discussion of best practices.

The tutorial will have the following format:

- Introductions: 5 Minutes
- Overview of CSG-Ed: 5 minutes
- CSG-Ed approaches: 10 minutes
- Breakout groups: 45 minutes

- Status of CSG on your campus
- Availability of local social-good projects on or near your campus
- Identification of possible locations for inclusion of CSG within your curricula
- Brainstorming about particular assignments or activities and CSG context
- Report out by group: 5 minutes
- Wrap up and Summary: 5 minutes

A series of prompts will be used to guide the discussion.

**Heidi Ellis** is a Professor of Computer Science and Information Technology at Western New England University. Her research area is student learning in humanitarian free and open source software (HFOSS) applications. She is a founding member of the HFOSS project and has been PI on five NSF grants related to HFOSS. She has over 15 years of experience supporting student learning via CSG. Heidi is one of the leads on the CS2023 Curricular Practices article on Computing for Social Good in Education.

**Gregory Hislop** is a Professor in the College of Computing and Informatics at Drexel University. He has been a Principal Investigator on five NSF grants to develop HFOSS education. He has developed HFOSS materials to include in existing courses and has also developed a complete HFOSS course. Gregory has broad experience in curricula development and has played a central role in the development of multiple computing degree programs. He is one of the leads on the CS2023 Curricular Practices article on Computing for Social Good in Education.

# Assignment Management and Automated Assessment Using Git-keeper

Conference Tutorial

Ben Coleman<sup>1</sup>, Nathan Sommer<sup>2</sup>

<sup>1</sup>Moravian University, Bethlehem, PA 18018

`colemanb@moravian.edu`

<sup>2</sup>Xavier University, Cincinnati, OH 45207

`sommern1@xavier.edu`

A multitude of systems exist to automatically test and provide feedback on student assignment submissions[4]. These tools are broadly defined by three aspects: distribution, collection, and assessment, and all tools have builtin assumptions about how these features should occur. In this tutorial, we will introduce git-keeper<sup>1</sup>, an assignment management system that automates distribution and collection of assignments using Git and provides feedback to students via email. In addition, it is based on the philosophy that automated feedback should encourage students to fix and re-submit their work. A second philosophical underpinning of git-keeper is that the assessment portion of assignment management should be configurable for each assignment.

During the tutorial, instructors will learn to leverage the capabilities of git-keeper in conjunction with BlueJ, a Java IDE designed for introductory courses. Participants will first act as students to complete a sample assignment using BlueJ. Once they understand the simplicity of git-keeper from the student perspective, participants will learn about the instructor workflow for managing assignments. We will demonstrate how git-keeper supports diverse types of assessments including unit tests, code linting, and any other command-line tool to generate feedback to the student. This makes git-keeper a tool that is suitable for a wide range of courses.

As a Git-based system, git-keeper provides students the ability to make multiple submissions, which fosters an iterative learning process and empowers them to refine their understanding and skills. It also provides a gentle introduction to basic Git workflows, easing the transition to using Git in software

---

<sup>1</sup><https://github.com/git-keeper/git-keeper>

engineering courses or the workplace. Students can clone starting repositories and push submissions using standalone Git clients or built-in IDE Git support, allowing instructors to choose the level at which students interact with the Git portion of the system's interface.

We have used git-keeper at our respective institutions for the past six years, and we have presented a poster[1], tutorial session[2], and workshop[3] about the tool. Instructors at three other schools have also used the tool successfully in courses ranging from CS1 to upper-level electives. While this tutorial will focus on integration with BlueJ, git-keeper is language and IDE agnostic, and we will discuss our experience using it in courses using Python, Jupyter Notebooks, Java, Kotlin, C, MIPS, or just about any other language.

This tutorial will give participants a hands-on experience using an instance of git-keeper on a cloud-hosted server, and provide links to online resources (example assignments, step-by-step instructions, etc.). To experience the student role, participants will need a laptop with BlueJ installed, which could be installed at the time of the tutorial.

## Biography

Ben Coleman and Nathan Sommer are the creators of git-keeper. As instructors at institutions with relatively small CS departments, they have taught and utilized git-keeper in numerous courses across the breadth of the CS curriculum, from introductory programming courses to upper level electives. They have also helped instructors at three other institutions adopt the system.

## References

- [1] Ben Coleman and Nathan Sommer. Git-keeper: An automated assignment testing system based on git. In *Proceedings of the 2016 Consortium for Computing Science in Colleges Eastern Conference*, Poster Presentation and Lightning Talk, 2016.
- [2] Ben Coleman and Nathan Sommer. Git-keeper: An automated assignment testing environment based on git. *Journal of Computing Sciences in Colleges*, 10 2018.
- [3] Ben Coleman and Nathan Sommer. Git-keeper: Streamlined software for automated assessment workflows. *J. Comput. Sci. Coll.*, 38(3):177, 11 2022.
- [4] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education (TOCE)*, 22:34, 6 2022.

# Using Continuous Integration (CI) For Managing Programming Assignments\*

Conference Tutorial

Sunjae Park  
School of Computing and Data Science  
Wentworth Institute of Technology  
Boston, MA 02115  
`parks6@wit.edu`

In this tutorial, instructors will be able to learn how to work with various tools on setting up a Continuous Integration Pipeline (CI pipeline), to automatically verify their programming assignments are valid. This tutorial assumes some programming experience.

Programming assignments with associated unit tests are a common type of assessment for many computing courses. One common method of distributing the programming assignment would be to use a version control system, such as git, and a version control website, such as GitHub or GitLab. The instructors post the project repository on the website, which the students then make a copy of (aka *clone*) [1, 2, 4].

A feature that many of these version control websites have is easy access to a CI pipeline. The CI pipeline allows programmers to run a variety of tasks automatically [3], but one common task is to checkout a project's source code into a clean system, build, and run unit tests and integration tests. However, it can also be used to run a variety of other tasks as well, such as running static analysis tools. These features are very helpful for computing instructors as well.

In this tutorial, I will discuss several basic tasks that the CI pipeline can do to help instructors. We will start with a basic programming assignment that is hosted on a source control website, and introduce various CI pipeline tasks one by one.

The first task to introduce is an automatic build task. It can be very confusing for students if the students first clone a programming assignment

---

\*Copyright is held by the author/owner.



and encounter build failures. The tutorial will explain how to set up a basic build task to automatically build the project in a clean environment, and send a notification (email) for any failures.

Next, a more advanced build task will next be introduced. Setting up a development environment can be a complicated task, with implicit settings and package installation that comes naturally to experts, but are challenging to novices. However, being able to install software is a necessary skill for all programmers, so instructors should try to make instructions as clear as possible. CI pipelines can help in this regard, since they always start from scratch in a clean environment.

Next, a set of tools will be introduced to verify that autograders work. Some programming assignments utilize unit tests and autograders. CI can be used so that instructors can provide multiple versions of the same assignment. The completed version should pass all tests, while the empty version (with just the skeleton code) should pass minimal tests.

Lastly, other maintainance tasks will be introduced. One such task is external package dependency analysis. Some assignments use external packages, and the package versions that the assignments were developed on can become outdated. A dependency analysis task can be set up so that the instructor gets notified when an external package becomes outdated. CI pipeline can automatically notify and sometimes even update the project for you.

## Biography

Sunjae Park started as an assistant professor at Wentworth Institute of Technology in Fall of 2020. Prior to joining Wentworth, he received his PhD in Computer Science from Georgia Institute of Technology and had worked at IBM as a operating system kernel developer.

## References

- [1] Miguel A Angulo and Ozgur Aktunc. “Using GitHub as a teaching tool for programming courses”. In: *2018 Gulf Southwest Section Conference*. 2019.
- [2] Nathan W Elo. “Teach like a git: streamlining assignment administration and enforcing good habits with professional tools and software development practices”. In: *Journal of Computing Sciences in Colleges* 36.6 (2021), pp. 27–36.
- [3] Martin Fowler. *Continuous Integration*. 2024. URL: <https://martinfowler.com/articles/continuousIntegration.html>.
- [4] Yu-Cheng Tu and et. al. “GitHub in the Classroom: Lessons Learnt”. In: *Proceedings of the 24th Australasian Computing Education Conference*. ACE '22. Virtual Event, Australia: Association for Computing Machinery, 2022, pp. 163–172. ISBN: 9781450396431. DOI: 10.1145/3511861.3511879. URL: <https://doi.org/10.1145/3511861.3511879>.

# A Process for Distinctive Curricular Design for Liberal Arts Computing Programs\*

Conference Tutorial

Jakob Barnard<sup>1</sup>, Grant Braught<sup>2</sup>, Janet Davis<sup>3</sup>,  
Amanda Holland-Minkley<sup>4</sup>, David Reed<sup>5</sup>, Karl Schmitt<sup>6</sup>,  
Andrea Tartaro<sup>7</sup>, James Teresco<sup>8</sup>

<sup>1</sup>University of Jamestown, Jamestown, ND 58405

`Jakob.Barnard@uj.edu`

<sup>2</sup>Dickinson College, Carlisle, PA 17013

`braught@dickinson.edu`

<sup>3</sup>Whitman College, Walla Walla, WA 99362

`davisj@whitman.edu`

<sup>4</sup>Washington & Jefferson College, Washington, PA 15317

`ahollandminkley@washjeff.edu`

<sup>5</sup>Creighton University, Omaha, NE 68178

`DaveReed@creighton.edu`

<sup>6</sup>Trinity Christian College, Palos Heights, IL 60463

`Karl.Schmitt@trnty.edu`

<sup>7</sup>Furman University, Greenville, SC 29690

`andrea.tartaro@furman.edu`

<sup>8</sup>Siena College, Loudonville, NY 12211

`jteresco@siena.edu`

The SIGCSE Committee on Computing Education in Liberal Arts Colleges (SIGCSE-LAC Committee) has found that liberal arts and small colleges approach design of their computing curricula in unique ways that are driven by institutional mission or departmental identity. This impacts how faculty at these colleges adopt curricular guidelines such as the current ACM/IEEE-CS

---

\*Copyright is held by the author/owner.

CS2013<sup>1</sup>. The committee is developing guidance, informed by its sessions at recent CCSC and SIGCSE conferences, to help with the design and/or revision of CS curricula in liberal arts contexts [1]. This will ultimately be included in the committee’s article in the Curricular Practices Volume that will be released as a companion to the new ACM/IEEE-CS/AAAI Computer Science Curricula guidelines (CS2023)<sup>2</sup>. Curricular guidelines like CS2013 or CS2023 inform curriculum design but are balanced with the vision for a program, departmental strengths, locale, student populations and unique academic experiences. The desire to craft distinctive curricula, combined with the size of prior curricular recommendations, requires an assessment of trade-offs between achieving full coverage of curricular recommendations and a school’s other priorities. SIGCSE-LAC’s guidance will encourage faculty to reflect on their programs and the role of CS2023, beginning with their institutional and departmental priorities, opportunities and constraints.

The specific goal of this session is to introduce participants to SIGCSE-LAC’s guidance to consider curricular development in the context of the unique features of their programs and . Following an overview and brief discussion of CS2023, participants will begin working through the latest version of the committee’s reflective assessment process. This process is framed by a series of scaffolding questions that begin from institutional and departmental missions, identities, contexts, priorities, initiatives, opportunities, and constraints. From there, participants will be led to identify design principles for guiding their curricular choices including the CS2023 recommendations. Examples gathered from the committee’s previous CCSC and SIGCSE sessions will be available to help to articulate identity and program design principles, which will then be used for the identification of identity-focused program-level learning outcomes. A spreadsheet tool that is being developed to aid in the shaping of curricular choices will be demonstrated. Participants will leave the session with a better understanding of how CS2023 can impact their programs and a jumpstart on the entire reflective assessment process. Feedback on the process and this session are welcome and will be used to refine the committee’s guidance prior to its publication in the CS2023 Curricular Practices volume.

## References

[1] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. Computer science curriculum guidelines: A new liberal arts perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 617–623, New York, NY, USA, 2023. ACM.

---

<sup>1</sup>[https://www.acm.org/binaries/content/assets/education/cs2013\\_web\\_final.pdf](https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf)

<sup>2</sup><https://csed.acm.org>

## Presenter Biographies

One of the eight co-authors of this session plans to serve as presenter.

**Grant Braught** is a Professor of Computer Science at Dickinson College. He is a facilitating member of the SIGCSE-LAC Committee, has organized committee events focused on curricula and has published widely on issues related to CS education, particularly within the liberal arts.

## Other Author Biographies

**Jakob Barnard** is Chair and Assistant Professor of Computer Science & Technology at the University of Jamestown. He is a member of the SIGCSE-LAC Committee and his research involves how curricula has been integrated into Liberal Arts Technology programs. **Janet Davis** is Microsoft Chair and Professor of Computer Science at Whitman College, where she serves as the department's founding chair. She co-organized SIGCSE pre-symposium events in 2020 and 2021 on behalf of the SIGCSE-LAC Committee. **Amanda Holland-Minkley** is a Professor of Computing and Information Studies at Washington & Jefferson College. Her research explores novel applications of problem-based pedagogies to CS education at the course and curricular level. She is a facilitating member of the SIGCSE-LAC Committee. **David Reed** is a Professor of Computer Science and Chair of the Department of Computer Science, Design & Journalism at Creighton University. He has published widely in CS education, including the text *A Balanced Introduction to Computer Science*, and served on the CS2013 Computer Science Curricula Task Force. **Karl Schmitt** is Chair and Associate Professor of Computing and Data Analytics at Trinity Christian College. He has served on the ACM Data Science Task Force and various Computing, Technology, Mathematics Education related committees for the MAA, ASA and SIAM. His interests explore data science education, and interdisciplinary education between computing, mathematics, data, and other fields. **Andrea Tartaro** is an Associate Professor of Computer Science at Furman University. Her computer science education research focuses on the intersections and reciprocal contributions of computer science and the liberal arts, with a focus on broadening participation. She is a member of the SIGCSE-LAC Committee, and has published and presented in venues including the CCSC and the SIGCSE Technical Symposium. **Jim Teresco** is Chair and Professor of Computer Science at Siena College. He has been involved in CCSC Northeastern for 20 years and currently serves as board chair, and has been involved with the SIGCSE-LAC Committee for 4 years. His research involves map-based algorithm visualization.

# Using the Accessible Learning Labs for Accessibility and Machine Learning Education\*

Conference Tutorial

Carla Lopez<sup>1</sup>, Kyle Messerle<sup>1</sup>, Samuel Malachowsky<sup>1</sup>,  
Daniel E. Krutz<sup>1</sup>, Jami Cotler<sup>2</sup>

<sup>1</sup>Department of Software Engineering  
Rochester Institute of Technology  
Rochester, NY 14623

{csl5623, klm3580, samvse, dxkvse}@rit.edu

<sup>2</sup>Department of Computer Science  
Siena College

{jcotler}@siena.edu

## 1 Abstract

Our *Accessible Learning Labs* (ALL) project informs participants about how to properly create accessible software and teaches participants about fundamental concepts of Artificial Intelligence and Machine Learning (AI/ML). To enhance readability, we will be using the abbreviation AI/ML to reference these concepts. These interactive learning modules demonstrate the need to create accessible software and provide hands-on experiences that showcase the multifaceted nature of AI's impact. This tutorial will benefit a wide-range of participants in the software engineering community, from students to experienced practitioners who want to further understand the implications of AI/ML in various domains and ensure that they are creating inclusive, accessible software. Complete project material is publicly available on the project website: <https://all.rit.edu>

---

\*Copyright is held by the author/owner.

## 2 Introduction

To fill the existing void in accessibility education, we have created *a comprehensive collection of laboratory activities that are essential to accessibility education*. Furthermore, in an aim to further AI/ML education, multiple educational labs have been created to understand the wider significance of AI/ML across different fields.

The labs are easy to integrate into a variety of existing introductory computing courses (*e.g.* Computer Science I & II) due to their easy to adopt, self-contained nature.

The labs and their experiential format have demonstrated the effectiveness in previous works [2, 3, 5, 1, 4].

### 2.1 Lab Structure

Each lab addresses at least one accessibility issue or AI/ML concept and contains: I) Relevant background information on the examined topic, II) An example app containing the accessibility problem or AI/ML topic, III) A process to emulate the problem (as closely as possible), IV) Testimonials from people on their real-life experiences with using non-accessible software or extra information regarding the AI/ML concept, and V) A quiz where the participant is tested on their acquired knowledge on the topic.

## 3 Tutorial Session Agenda

**Activity 1: Ethics of AI-focused Lab: (30 minutes)** This lab introduces participants on the ethics behind the implementation of Artificial Intelligence models. This lab guides users through multiple scenarios where the AI is biased.

**Activity 2: Literacy-focused Lab: (30 minutes)** This lab instructs participants on proper procedures in making software accessible to users with different literacy levels. Participants learn about the importance of creating comprehensive and readable web pages through the Fog Index formula.

**Activity 3: Lab Feedback: (15 minutes)** Participants will provide feedback on the material, and offer guidance to presenters on the future direction of the labs. This feedback will be incorporated into the design of future labs.

### Acknowledgements

This material is based upon work supported by the NSF under grant #1825023, #2145010, and #2111152.

## References

- [1] Yasmine El-Glaly et al. “Presenting and evaluating the impact of experiential learning in computing accessibility education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. 2020, pp. 49–60.
- [2] Saad Khan et al. “Experiential learning in undergraduate accessibility education: Instructor observations”. In: *Journal of Computing Sciences in Colleges* 38.8 (2023), pp. 54–68.
- [3] Heather Moses et al. “Experiential Educational Accessibility Modules”. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–3.
- [4] Weishi Shi et al. “ALL: Supporting Experiential Accessibility Education and Inclusive Software Development”. In: *ACM Transactions on Software Engineering and Methodology* 33.2 (2023), pp. 1–30.
- [5] Weishi Shi et al. “Experiential learning in computing accessibility education”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. 2020, pp. 250–251.

# Assessing Risks, Challenges and Opportunities of Generative AI in Computer Programming Education\*

Lightning Talk

Sa Liu<sup>1</sup>, Brian Grey<sup>2</sup>, Ryan Watkins<sup>3</sup>,  
Chad Chu<sup>2</sup>, Phillip Grim<sup>2</sup>, Thomas McManus<sup>4</sup>

<sup>1</sup> Interactive Media

<sup>2</sup> Computer and Information Sciences Program

<sup>4</sup> Center for Innovative Teaching and Learning  
Harrisburg University

<sup>3</sup> George Washington University

saliu@harrisburg.edu

Artificial Intelligence (AI) has the potential to transform the education sector by enhancing teaching and learning experiences. According to Sal Khan, founder of Khan Academy, AI is about to start "the biggest positive transformation that education has ever seen"<sup>1</sup> by making high-quality personalized tutoring available (tuition free) to everyone on the planet. Given AI's, and more specifically Generative AI's (GAI), rapidly developing capabilities (e.g., to provide tailored feedback, ask questions of students, give examples and non-examples, and offer general learning support), incorporating GAI into programming education has the potential to enhance student engagement and learning outcomes. At the same time, they identified challenges in using GAI, such as its inability to answer some questions and its tendency to provide incorrect or incomplete responses. Students also report an increase in anxiety surrounding GAI and its potential effects on future professional opportunities. Outside of the classroom there is likewise an increasing prevalence of GAI in computational professions, making it crucial to equip students with the necessary knowledge and skills to effectively, responsibly, and ethically utilize GAI.

---

\*Copyright is held by the author/owner.

<sup>1</sup><https://blog.khanacademy.org/sal-khans-2023-ted-talk-ai-in-the-classroom-can-transform-education/>



Rather than avoiding the use of GAI in the classroom, in this study we aim to investigate the pros and cons of leveraging GAI's capabilities to offer personalized guidance and assistance to students as they learn programming. By doing this research, we are learning to create more interactive and engaging learning experiences that better equip students with the skills and knowledge needed to succeed in the field of programming. This project, which is currently being conducted, was designed to address this research question: To what extent does the incorporation of GAI impact students' engagement, motivation, and achievement, particularly with the material in Intro to Programming courses and their chosen STEM field of study? It is utilizing case studies that focus on the integration of GAI into computer programming education. The team has 1) developed a series of GAI-supported teaching modules specifically designed to improve problem-solving skills in programming tasks among undergraduate students; and 2) is in the process of analyzing student feedback on GAI integration in computer programming education. This project offers an important exploration into the intersection of GAI and programming education, with the expectation that results will provide useful guidance for programming instructors who are adapting their instructional strategies for the emerging role of GAI in programming. The team will briefly present the status of the research and early insights from the project, and then engage with the audience on how lessons learned from this work can pragmatically shape programming courses in their institutions. Quick tips, takeaways, and prompting strategies will be shared throughout this interactive lightning talk.

# Unambiguous Non-Functional Requirements Specification – An Extended Use-Case\*

Lightning Talk

Unnati Shah<sup>1</sup>, Krupa Patel<sup>2</sup>

<sup>1</sup>Department of Computer Science

Utica University, NY 13502

[unshah@utica.edu](mailto:unshah@utica.edu)

<sup>2</sup>PhD Scholar

Bhagwan Mahavir University, India 395007

[krupa.3112@gmail.com](mailto:krupa.3112@gmail.com)

This past summer, I worked with a PhD scholar (Krupa Patel, Bhagwan Mahavir University, India) on a research project that investigates a novel approach for specifying unambiguous Non-Functional Requirements (NFRs). NFRs are usually written in common natural language. Due to differences in the domain expertise of requirements engineers and other stakeholders in the project, it is possible that the NFRs contain several words that allow alternative interpretations. To improve trust between clients and software organizations, the following problems need to be addressed at an early stage of software development:

- Recognizing words that are ambiguous in NFRs.
- Determining conflicts related to the NFRs.
- Providing a precise, in-depth understanding of the NFRs by specifying conflicting NFRs.

All the above-mentioned problems are crucial and should not be delayed because software requirements (functional and non-functional) act as a legal agreement between the software organization and clients. Our goal in this research project is to examine approaches for recognizing ambiguous words,

---

\*Copyright is held by the author/owner.

conflicts among NFRs and visualize them using extended Unified Modelling Language (UML) models viz. extended use-case diagram. The extended use-case diagram will help requirements engineers make decisions early in the software development process by helping them visualize the effects of competing NFRs on each other. The proposed approach addresses problems such as normalization, resolution of ambiguity, and clustering of NFRs using efficient semi-formal approaches (viz. NLP and ML) and formal approaches (viz. ontology). The empirical evaluation of the proposed approach on the publicly accessible dataset PROMISE achieves an average result of 79.76% recall, 90.05% precision, and 84.59% F-measure. The findings of the assessment suggest that the deployment of the approach will have a positive effect on the practice of RE. Nevertheless, we find issues with the validity of the proposed approach, such as the fact that NFRs can be classified using one or more NFR attributes, apart from use-case diagrams, which are other UML models that are best suited for specifying NFRs early in the software development process and which documents improve NFR classification performance. We need constructive criticism from academic and industry professionals to help us improve the design of the proposed approach.

**Biography:** Unnati Shah has been working as an assistant professor in the department of computer science at Utica University, NY, US. She received her Ph.D. in Computer Science and Engineering from the S. V. National Institute of Technology, India in 2022. Her research interests span both Software Engineering and Natural Language Processing. Much of her work has been on software requirements engineering, especially resolving ambiguities from natural language software requirements, and providing formal specification.

# Redesigning Introduction to Programming Course Work using a Project-based Learning Approach\*

Lightning Talk

Ali Al-Faris, Saba Kadady  
Computer Science Department  
Worcester State University, Worcester, MA 01602  
`{aalfaris,skadady}@worcester.edu`

While teaching Introduction to Programming in the Computer Science programs for many semesters, it was noticed that the students were not necessarily able to see the big picture of what they were learning and where it could be applied. It was also noticed from meetings with individual students during office hours and beyond that students, after completing the course, continued to feel hesitant to tackle big projects because they had not worked on them before. That was the main motivation to modify the way coursework was designed. The term project gives students a chance to implement real-world problems and apply the course's material to larger tasks. The project includes multiple parts. Each part is assigned after completing a new topic(s) and its labs. Each subsequent part builds on and expands the preceding work. The project requires students to understand the concepts and problem specifications, design and implement the solution, test, and debug it.

The importance of adopting this approach (one big project divided into parts) as opposed to traditional individual, unrelated assignments is evident in students' interest in working on real-world projects that they can relate to and have seen to some extent in their daily lives. Also, completing this project gives students the experience of working on large-scale projects that require multiple and continuous modifications, updates, and refactoring of the current version of the project. Each student is individually assigned one of three themes for a project. Each theme is composed of seven main parts. Each part covers a set of subtopics covered in lectures and labs.

---

\*Copyright is held by the author/owner.

The three themes used in this semester are: Health Record System, Day Care Management System and Restaurant Management System.

To investigate the effectiveness of this approach to course work redesign, the study consists of online, anonymous surveys. Students who choose to participate take the surveys anonymously distributed after completing each part towards building the final project.

# Exercises on Distributed List Operations\*

## Lightning Talk

Jan Plaza

Computer Science Department, SUNY Plattsburgh  
101 Broad Street, Plattsburgh, NY 12901, USA  
[jan.plaza@plattsburgh.edu](mailto:jan.plaza@plattsburgh.edu), [github.com/plazajan](https://github.com/plazajan)

**Keywords:** distributed list, distributed computing, line network, simulation, Python 3, complexity, domino, activity diagram, educational.

## 1 Introduction

Our program simulates a chain (a case of line network) of asynchronous processing elements (PEs), each communicating only with its neighbors. The chain can store a list of integers, one integer per PE. This provides an environment for 40 exercises on design and implementation of distributed algorithms for list operations, from deque operations to sorting, and on evaluating their complexity. To aid students in the complexity analysis we introduce visualizations called Dominos and Activity Diagrams. The algorithms require a different design and may have complexity different from their counterparts on arrays or linked lists. They can be run on an any computer with Python 3 and offer students the familiar syntax of that language.

## 2 Code, Dominos, Activity Diagrams, and Complexity

Students will write code similar to the following, for testing if an integer received via `connectorUpper` is in the list. Communications commands with a suffix `_w` are for a word, i.e. integer, and `_b` for a bit. `send_o` tells the lower PE to execute `member` code.

---

\*Copyright is held by the author/owner.

```

def member(self, channelA):
    if not self.bit[channelA]: # if storage channelA is non-empty
        self.temp_w[0] = self.connectorUpper.receive_w()
        if self.word[channelA] == self.temp_w[0]: # found here
            self.connectorUpper.send_b(True)
        else: # keep looking below
            self.connectorLower.send_o("member", channelA)
            self.connectorLower.send_w(self.temp_w[0])
            self.temp_w[0] = self.connectorLower.receive_b()
            self.connectorUpper.send_b(self.temp_w[0])
    else: # if current PE terminates the list
        _ = self.connectorUpper.receive_w()
        self.connectorUpper.send_b(False) # not found anywhere

```

PEs involved in the computation execute a sequence of communications of types represented by the dominos in Figure 1. The top and the middle dominos are the same, except that the former is stretched horizontally, because of the passage of time.

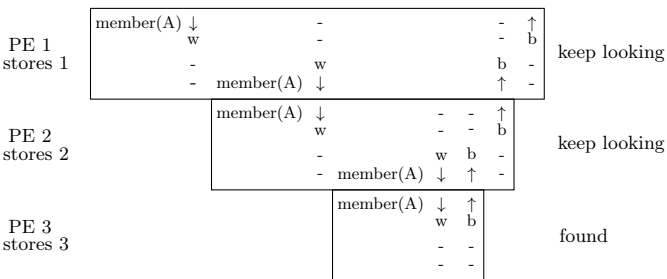


Figure 1: Domios in an activity diagram - looking for 3 in a list [1,2,3,4]

For the `member` operation, on a list of length  $n$ , in the worst case (when the item is not in the list), the activity diagram such as that in Figure 1, has width  $O(n)$  leading to the conclusion that overall time complexity of `member` is  $O(n)$ . We also introduce the concept of top complexity, which can be thought of as the width of the top domino; in the case above the top complexity is  $O(n)$  as well.

Another exercise, on selection sort, has an interesting activity diagram.

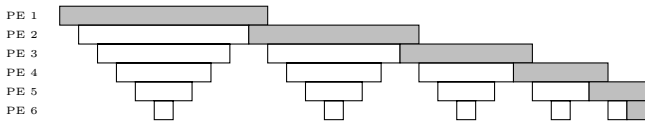


Figure 2: Selection sort - activity diagram for a 5-element list

The first inverted triangle on the left, brings the smallest value to PE 1. The second triangle brings the second smallest value to PE 2, ..., the  $n$ -th triangle brings the  $n$ -th smallest value to PE  $n$ . After making such diagrams, the students will see that the top complexity of the algorithm is  $O(n)$ , where  $n$  is the length of the list, and the overall time complexity is  $O(n^2)$ .

# An Accelerated Java Programming Summer Program\*

Poster Abstract

Ching-yu Huang and Paolien Wang  
Department of Computer Science & Technology  
Kean University, Union, NJ 07083  
{chuang, pawang}@kean.edu

Since 2018, New Jersey high schools have been required to offer computer science courses, and the nature of these courses can vary significantly. This proposal suggests implementing an accelerated Java summer program specifically designed for incoming new students at colleges. The freshman class has expanded to include many students with no coding background, and some have a limited background but are either unwilling or unable to take the AP Computer Science exam. Consequently, these students often enroll in a full semester CS1 course as their first programming course at college. However, an accelerated Java summer program would better serve some students, building upon their existing knowledge. This program allows freshman students to advance to the second programming class in their first semester, instead of being mixed with other students with no coding background. This proposed program represents a departure from the conventional 4-credit Java section, compressing what is typically a 15-week course (4 hours per week, comprising 3 hours of lecture and 1 hour of lab) into an intensive 2-week format. The success of this program depends on the invited students' prior programming experience, motivation, and their ability to quickly grasp the curriculum, allowing them to thrive in a fast-paced learning environment. The development of a cohort community of students through this shared experience is expected to positively impact student retention and success.

The program will be divided into a morning session dedicated to lectures, covering the theoretical aspects of the course, and afternoon sessions focusing on hands-on experiences, debugging skills, and the interrelation of various top-

---

\*Copyright is held by the author/owner.



ics. Additionally, each day will allocate 1.5 hours for students to work with instructive supervision on lab assignments during the afternoon.

Following the textbook 'Introduction to Java Programming and Data Structures' by Y.D. Liang (12th Edition), the curriculum will cover Chapter 1 - Introduction to Computers, Programs, and Java; Chapter 2 - Elementary Programming, Selections; Chapter 3 - Selections; Chapter 4 - Mathematical Functions, Characters, and String; Chapter 5 - Loops; Chapter 6 - Methods; Chapter 7 - Single-Dimensional Arrays. Given that the selected students are expected to have some background in programming, the curriculum will focus more on advanced topics, problem-solving, and debugging within Chapters 4, 5, 6, and 7.

Keywords - Accelerated, AP, CS1, Java, Programming Language, Summer Program.

# Code Samurai Program: Empowering Students in Computer Science and Technology\*

Poster Abstract

Paolien Wang, Ching-yu Huang and Patricia Morreale  
Department of Computer Science & Technology  
Kean University, Union, NJ 07083  
{pawang, chuang, pmorreale}@kean.edu

Over the past decade, there has been an increase in enrollment in computer science-related programs at colleges and universities. This surge can be attributed to the expansion of the technology industry during this period and the need for a technologically knowledgeable workforce, resulting in a diverse influx of students with varying levels of expertise and knowledge in computer science.

The increasing interest in computer science mirrors the industry's growth, signaling a heightened awareness of technology's pivotal role in various sectors. With this heightened enrollment, the student body now comprises individuals boasting a wide spectrum of skills and backgrounds, fostering a dynamic and diverse learning environment.

However, this surge in enrollment has presented challenges, particularly in the realm of academic support. Many students, regardless of their varied levels of experience, need assistance with homework assignments. The demand for support underscores the significance of providing effective resources and support systems to ensure that students can navigate their coursework successfully and capitalize on their educational experience in the rapidly evolving field of computer science.

Securing resources for teaching assistants in every Computer Science and Technology program class poses a substantial challenge for many schools. This predicament requires significant funding and necessitates a sizable pool of enthusiastic and highly skilled students each semester. For commuter schools, where most students commute daily rather than residing near the campus, recruiting an adequate number of teaching assistants becomes even more formidable.

---

\*Copyright is held by the author/owner.

Moreover, students in these universities often juggle part-time or full-time jobs outside the campus, amplifying the complexity of providing effective support for assignment troubleshooting and major course studies. Without supportive peer-learning communities, commuter and working students are at a disadvantage in their computing studies and may not be able to participate in the professional fields they aspire to.

In response to these challenges, the Code Samurai program, a peer-learning peer mentoring program, has demonstrated successful implementation over a decade, supporting a rapidly growing department. This program's adaptability and scalability make it a potential solution for colleges facing similar resource constraints. This paper presents a comprehensive model outlining the recruitment, management, and tutoring processes involved in the Code Samurai program, illustrating the benefits to the students seeking learning reinforcement, as well as the peer leaders themselves. The aim is to offer insights and guidance for institutions seeking effective strategies to address the unique challenges of supporting students in computer science and technology programs in regional universities.

Keywords – Code Samurai Program, Coding Assignments, Computer Science, Information Technology, Problem-Solving, Regional Universities, Teaching Assistantship, Tutoring

# Tech Startup: A Multi-Level Multi-Semester Project Course \*

Poster Abstract

Kevin McCullen  
Computer Science Department  
State University of New York at Plattsburgh  
Plattsburgh, NY 12901  
kmccu006@plattsburgh.edu

Project Based Learning involves students working on projects that reflect realistic problems [2]. The standard ACM curriculum contains a number of references to projects and group work [1]. In project based courses, students are typically either assigned to small groups, or self-organize into groups. Each group undertakes one or more software projects that significantly exceed a typical homework type problem.

Our Computer Science curriculum includes one required purely experiential course, the Information Technology curriculum includes two, and the Robotics curriculum has one such course each year.

We have introduced a new experiential course sequence called the Tech Startup. The course sequence includes three courses; one at the 200 level, one at the 300 level, and one at the 400 level. Each semester, all three are offered, and they meet as one larger course. Both the credits assigned and the responsibilities and expectations differ for each section. The 200 level course is referred to as a "Junior Developer", and is a one credit course. The 300 level course is the "Developer" course, and is a two credit course. The 400 level course is called "Team Lead", and is a three credit course. Students may retake each of the courses.

Students are admitted by permission of instructor, and typically will need to apply for admission (using a Google Form to collect a resume). The team leads in particular meet with the faculty member in order to determine their

---

\*Copyright is held by the author/owner.

suitability for the role, and their thoughts and plans for a project.

In this poster, we present our experiences designing and offering this new experiential course model. Our poster will deal with how the course was managed and students assessed, the particular skills required of the instructor, the student outcomes (both in terms of projects, and as assessed by Course Opinion Surveys), and our experience in offering the courses. The course sequence was first offered in the Spring of 2020, and this poster describes experiences and outcomes through the Fall of 2022 (five semesters).

We want to particularly acknowledge Dr. Delbert Hart, who initially conceived of the idea of this course sequence and who was very supportive in the initial offerings.

## References

- [1] ACM Curriculum Recommendations: Computing Curricula 2020, 2020.
- [2] Michael J. O'Grady. Practical Problem-Based Learning in Computing Education. *ACM Transactions on Computing Education*, 12(3):1–16, July 2012.

# Beyond the Pixels: An Interdisciplinary Approach to Image Analysis in Astronomy\*

Poster Abstract

Nicole Granucci, Chetan Jaiswal  
Colleges of Arts and Science  
School of Computing and Engineering  
Quinnipiac University  
Hamden, CT 06518

{Nicole.granucci,Chetan.jaiswal}@quinnipiac.edu

Interdisciplinary collaboration between computer science and astronomy is critical for new discoveries as it brings together the expertise of two distinct yet complementary fields. The vast and complex datasets generated by astronomical observations require sophisticated computational methods for analysis and interpretation. Computer scientists contribute their skills in algorithm development, data processing, and artificial intelligence whereas astronomers provide unique challenges and insights that push the boundaries of computer science applications. This collaboration offers a multifaceted learning experience for students beyond disciplinary boundaries. This approach encourages a diverse skill set development, promotes creativity, critical thinking, and adaptability. It provides students with real-world applications for their theoretical knowledge, preparing them for careers that demand cross-disciplinary collaboration. It also instills a collaborative mindset that is increasingly essential in today's interconnected scientific landscape. Our intent is to develop a baseline model and then involve students during the summer research experience program. We also foresee developing a Computer Science-Physics cross listed undergraduate course for enriching student experience who are interested in these two disciplines. In this study, we will present our work in progress towards developing open-source software for astronomy image study and analysis. The flexibility of open-source development combined with a culture of innovation, leads to rapid improvements and the introduction of cutting-edge features. This soft-

---

\*Copyright is held by the author/owner.

ware would provide a transparent mode for non-programmer users to not only view, stack, and overlay the raw image capture(s) but also do basic photometry and image statistics. The work in progress also involves the study of unsupervised learning to perform clustering analysis of the raw image capture(s). This analysis would group pixels or regions with similar characteristics, uncovering structures such as globular clusters, galaxies, or binary star systems. This process helps discern spatial relationships, identify outliers, and distinguish different celestial objects based on their inherent properties, such as brightness and shape. We are testing our open-source software with image data from a college telescope from Southern Connecticut State University. The telescope is a Spica Eyes Dobsonian reflecting telescope with an aperture size of 0.6m and focal ratio f/3.3. Transfer optics were added to increase our spatial resolution from 1.4 arcsec per pixel to 0.8 arcsec per pixel measurement. The transfer optics used a filter  $\lambda_0 = 537\text{nm}$  that optimized quantum efficiency in the V-band. These images were taken July 2018 at Southern Connecticut State University (SCSU) in New Haven, CT where there is high artificial brightness and Bortle scale of 8. These images were taken in poor seeing conditions since the humidity was high due to the typical nature of coastal Connecticut in the summer. Furthermore, tracking was not stable leading to target drift across the image plane. Figure 1(a) shows images from this observation session which include 8 frames of 0.05s exposure time with a readout speed of 2MHz of Jupiter. This shows how tracking and poor alignment can lead to images that are not resolved to show the details of Jupiter as seen in Figure 1(b). Stacking is necessary to improve the signal to noise ratio of the images as well as smooth out details. Without sufficient stacking, it will be difficult to distinguish details of the images.

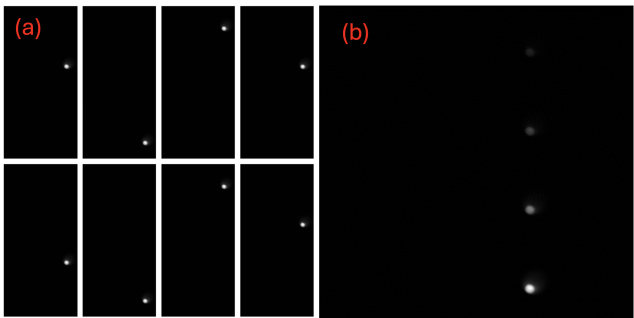


Figure 1: (a) 0.05s exposure frames of Jupiter. (b) Images of Jupiter stacked.

One way to overcome poor tracking is a method called Shift and Add which was first introduced by Bates and Cody in 1980 [1]. This process uses the cen-

centroid pixel of the object of interest on one image. Then finds the same object of interest and its centroid pixel location. The two images are then stacked upon on another based on their centroid locations, thus the images are shifted and then stack on top of one another. This can be done by hand if you have a few images, but generally for best image resolution, you need to take at least 30 stacks to overcome signal to noise. Using a computer automated shift and add stacking ability will significantly increase image quality and reduce image processing time for poor tracking conditions. Mizar was another target captured at SCSU that evening we used to test our image rendering and modeling using computer algorithms. Mizar A is an A2V star with V magnitude of 2.20 located in the constellation Ursa Major was chosen for its high V magnitude and ability to be seen by the naked eye. It is a visual double with Alcor, an A5V star with V magnitude of 4.01 which is a gravitational bound binary 706 arcsec apart. However, Mizar has a closer companion 14.42 arsec away called Mizar B, a kA1h(eA)Ma7IV-V star that is a spectroscopic binary with V magnitude 3.88 that can only be seen using a telescope [3]. Figure 2(a) shows an image of Mizar and its faint companion to the bottom right. Figure 2(b) shows its intensity plot across the central axis of Mizar. Stars should model a dot on the image plane and a smooth intensity profile due to circular aperture of the telescope also known as a point spread function. However, Figure 2(a) demonstrates the poor collimation of the telescope with v-like shape coma. This makes it difficult to resolve the star into its theoretical point source profile. Figure 2(b) displays the intensity profile with jagged lines unlike the smooth peak a tradition point spread function would demonstrate.

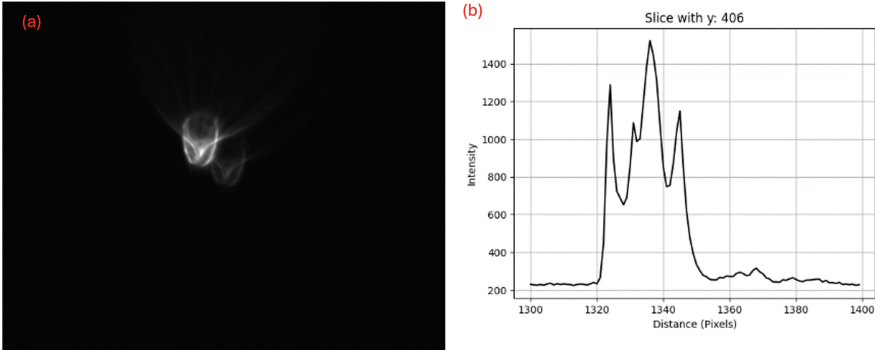


Figure 2: (a) 0.01 exp of Mizar A and Mizar B (b) Seeing Profile of Mizar A.

This collaboration explores the challenges astronomers face with image processing with solutions from computing. We explore the initial steps to tackling



some of these astronomy initiatives by developing new algorithms in python. Previous research from Granucci, et al. [2] used IDL to render images and perform image analysis. However, python is more widely available to students, therefore we aim to use that platform for image analysis. We aim to offer summer research experience and a cross listed course in computing and astronomy to develop new algorithms to be used on images taken at SCSU for calibration. Once we have tested our system, we will take new images with our ZWOASI294 Pro Color Camera and Explorer Scientific refractor telescope,  $f/7$ , at Quinnipiac University to study our different seeing conditions as well as optics for image research.

## References

- [1] RHT Bates and FM Cady. “Towards true imaging by wideband speckle interferometry”. In: *Optics Communications* 32.3 (1980), pp. 365–369.
- [2] Nicole M Granucci and Elliott P Horch. “Wide-field speckle techniques for small, urban telescopes”. In: *Optical and Infrared Interferometry and Imaging VI*. Vol. 10701. SPIE. 2018, pp. 279–286.
- [3] Mizar. *SIMBAD*. Centre de données astronomiques de Strasbourg. Retrieved 2024-02-09.

# Redesigning Introduction to Programming Course Work using a Project-based Learning Approach \*

Poster Abstract

Ali Al-Faris, Saba Kadady  
Computer Science Department  
Worcester State University  
Worcester, MA 01602  
{aalfaris, skadady}@worcester.edu

While teaching Introduction to Programming in the Computer Science programs for many semesters, it was noticed that the students were not necessarily able to see the big picture of what they were learning and where it could be applied. It was also noticed from meetings with individual students during office hours and beyond that students, after completing the course, continued to feel hesitant to tackle big projects because they had not worked on them before. That was the main motivation to modify the way coursework was designed. The term project gives students a chance to implement real-world problems and apply the course's material to larger tasks. The project includes multiple parts. Each part is assigned after completing a new topic(s) and its labs. Each subsequent part builds on and expands the preceding work. The project requires students to understand the concepts and problem specifications, design and implement the solution, test, and debug it.

The importance of adopting this approach (one big project divided into parts) as opposed to traditional individual, unrelated assignments is evident in students' interest in working on real-world projects that they can relate to and have seen to some extent in their daily lives. Also, completing this project gives students the experience of working on large-scale projects that require multiple and continuous modifications, updates, and refactoring of the current version of the project. Each student is individually assigned one of three themes for a

---

\*Copyright is held by the author/owner.

project. Each theme is composed of seven main parts. Each part covers a set of subtopics covered in lectures and labs.

The three themes used in this semester are: Health Record System, Day Care Management System and Restaurant Management System.

To investigate the effectiveness of this approach to course work redesign, the study consists of online, anonymous surveys. Students who choose to participate take the surveys anonymously distributed after completing each part towards building the final project.

# Automating Paper-based Peer Reviews\*

Poster Abstract

Delbert Hart  
Computer Science Department  
SUNY Plattsburgh  
Plattsburgh, NY 12901  
hartdr@plattsburgh.edu

Many colleges have advanced writing requirements for all students, including computer science majors. At our college one of the requirements that we must meet is to have a robust peer review process. This poster describes a set of custom software tools used to support double-blind peer reviews of handwritten student papers.

The primary motivation to requiring handwritten essays is as a counter-measure against generative AI usage. While estimates vary widely, from 10%[1] to 56%[6], it is a factor that needs to be addressed in any writing course. Although requiring handwriting does not prevent generative AI usage, it has been effective at reducing its impact. The downside of using paper though is that it makes double-blind peer review more cumbersome.

The tools used to support the review process are all open-source tools and libraries tied together with a set of Perl scripts. The first pair of tools used are Libqrencode[5] and ZBar[3] to create and read, respectively, QR codes for tracking a paper's author. The instructor hands out papers marked with the student's name and a unique QR code. Assignments can either be full page open-ended writing, or can be a set of short answer and/or multiple choice questions. Review sheets are likewise marked, and may either be full page, or there may be multiple reviews on the same page. The data embedded in the QR codes is the assignment number and a random assignment-specific token.

An SQLite[4] database file is used to track documents, any subregions of documents, review assignments, and any associated meta-data.

Opencv[2] is used to remove the name and QR codes before redistributing information to either reviewers or reviewees. An earlier version of these tools

---

\*Copyright is held by the author/owner.

used pdftjam to simply crop the documents based on the expected position of different pieces of information. This mostly worked but could fail based on irregularities that might occur from either printing or scanning. The current version of the software uses Opencv to look at the contours present in the document and then base editing decisions on that information. It can also be used to recognize responses to multiple choice questions. Opencv is used to split documents into subregions and then those subregions can be selectively reintegrated into documents for the other students or the instructor. For instance, a review sheet can have an area for a reviewer to include comments to the instructor that are not shared with the reviewee.

The recombination feature is also useful for collecting feedback on in-class presentations and redistributing the feedback to the presenters.

Documents are created using a script that generates LaTeX files. The LaTeX files include the images that the Opencv scripts cropped. The LaTeX files are compiled into pdfs worksheets that are printed and distributed to the class. The documents that students review are uploaded to a google drive folder, and then an Apps Script program distributes the review assignments into folders shared with each individual student.

The total size of the software is about 500 SLOC of Perl and Python scripts. The software handles identification, anonymization, randomized review assignments, and feedback collation tasks. The instructor is responsible for distributing the paper worksheets and scanning them when they are turned in. Camera based submissions do also work, but scanning provides better uniformity ( and usually better quality due to lighting ). These tools have made double-blind peer reviews relatively simple to implement for class assignments.

## References

- [1] anthology.com. Ai in higher ed: Hype, harm, or help. Technical report, Anthology Inc., 2023.
- [2] Gary Bradski, Adrian Kaehler, et al. Opencv. *Dr. Dobb's journal of software tools*, 3(2), 2000.
- [3] Jeff Brown. Zbar. <https://github.com/mchehab/zbar>, 2024.
- [4] SQLite Consortium. Sqlite. <https://www.sqlite.org/>, 2024.
- [5] Denso Wave Inc. Libqrencode. <https://github.com/fukuchi/libqrencode>, 2020.
- [6] Jane Nam. 56% of college students have used ai on assignments or exams. <https://www.bestcolleges.com/research/most-college-students-have-used-ai-survey/>, 2023.

## Reviewers — 2024 CCSC Northeastern Conference

|                      |   |
|----------------------|---|
| Chris Alvin          | Furman University                       |
| Jami Cotler          | Siena College                           |
| Lawrence D’Antonio   | Ramapo College                          |
| Dan DiTursi          | Siena College                           |
| Karyn Doke           | University at Albany                    |
| Peter Drexel         | Plymouth State University               |
| Razieh Fathi         | Whitman College                         |
| Timothy Fossum       | Rochester Institute of Technology       |
| Martin Gagne         | Wheaton College                         |
| Alessio Gaspar       | University of South Florida Polytechnic |
| Michael Gousie       | Wheaton College (MA)                    |
| Nadeem Hamid         | Berry College                           |
| Delbert Hart         | SUNY Plattsburgh                        |
| Susan Imberman       | City University of New York             |
| Stella Kakavouli     | Wellesley College                       |
| Sotirios Kentros     | Salem State University                  |
| Bo Kim               | Southern New Hampshire University       |
| Richard Kline        | Pace University                         |
| Daniel Krutz         | Rochester Institute of Technology       |
| David Levine         | Saint Bonaventure University            |
| Zhengping Luo        | Rider University                        |
| Sriharsha Mallapuram | Plymouth State University               |
| Stephanos Matsumoto  | Olin College of Engineering             |
| Robert McCloskey     | University of Scranton                  |
| Kevin Mccullen       | SUNY Plattsburgh                        |
| Mehdi Mekni          | University of New Haven                 |
| Sunjae Park          | Wentworth Institute of Technology       |
| Sofya Poger          | Felician University                     |
| Rajesh Prasad        | Saint Anselm College                    |
| Daniel Rogers        | The College at Brockport                |
| Unnati Shah          | Utica University                        |
| Ashley Suchy         | SUNY New Paltz                          |
| William Tarimo       | Connecticut College                     |
| James Teresco        | Siena College                           |
| Scott Valcourt       | Northeastern University                 |
| Nanette Veilleux     | Simmons College                         |
| Marc Waldman         | Manhattan College                       |
| Yang Wang            | La Salle University                     |

