The Journal of Computing Sciences in Colleges

Papers of the 27th Annual CCSC Northwestern Conference

October 10th and October 11th, 2025
Saint Martin's University
Lacey, WA

Abbas Attarwala, Editor
California State University, Chico

Sharon Tuttle, Regional Editor
Cal Poly Humboldt

Volume 41, Number 1 **October 2025**

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.

Table of Contents

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:

Bryan Dixon, President (2026), bcdixon@csuchico.edu, Computer Science Department, California State University Chico, Chico, CA 95929.

Shereen Khoja, Vice President/President-Elect (2026), shereen@pacificu.edu, Computer Science, Pacific University, Forest Grove, OR 97116.

Abbas Attarwala, Publications Chair (2027), aattarwala@csuchico.edu, Department of Computer Science, California State University Chico, Chico, CA 95929.

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

Haiyan Cheng, Membership Secretary (2028), hcheng@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.

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.

Gabe Ferrer, Midsouth Representative (2028), ferrer@hendrix.edu, Department of Computer Science, Hendrix College, Conway, AR 72032.

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

Michael Gousie, Northeastern Representative (2028), gousie_mike@wheatoncollege.edu, Computer Science Department, Wheaton College, Norton, MA 02766.

Ben Tribelhorn, Northwestern Representative (2027), tribelhb@up.edu, School of Engineering, University of Portland, Portland, OR 97203.

Mohamed Lotfy, Rocky Mountain Representative (2028), MohamedL@uvu.edu, Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Mika Morgan, South Central Representative (2027), mika.morgan@msutexas.edu, Department of Computer Science, Midwestern State University, Wichita Falls, TX 76308.

Karen Works, Southeastern Representative (2027), keworks@pc.fsu.edu, Department of Computer Science, Florida State University Panama City, Panama City, FL 32405.

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

Serving the CCSC: These members are serving in positions as indicated:
Bin "Crystal" Peng, Associate Editor, bin.peng@park.edu, Department of Computer Science and Information Systems, Park University, Parkville, MO 64152.
Lucas Cordova, Associate Treasurer, lpcordova@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.
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.
Ed Lindoo, UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.
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
Blossoms

Welcome to the 2025 CCSC Northwestern Conference Saint Martin's University

On behalf of the Conference Steering Committee and the Northwest Regional Board, it is sheer pleasure to extend a very warm welcome to all attendees of the Twenty-Seventh Annual Consortium for Computing Sciences in Colleges (CCSC) Northwestern Regional Conference. We are honored to host this year's event at Saint Martin's University and are eagerly looking forward to engaging with you all in person.

I would like to extend my sincere gratitude to the many individuals and groups who contributed their time and effort to coordinating and supporting this year's conference. Their dedication and drive have been impressive. This year's program features 11 peer-reviewed papers, 2 tutorials, 2 partner presentations, and a keynote address by Dr. Ameeta Agrawal, Director of the NLP Lab and Assistant Professor at Portland State University, who will deliver a talk on a timely topic; "LLMs and the Shifting Landscape of CS Education". All submissions underwent a rigorous peer review process, with 11 out of 17 papers being accepted, resulting in an acceptance rate of 65%. Additionally, we are thrilled to have received 13 exceptional student poster submissions, each of which will be presented at the conference.

We are deeply grateful to our national and local partners, without whom this conference would not be possible. Above all, we thank you, the attendees, whose presence here strengthens the vital networks of communication and collegiality that sustain and advance our discipline.

It is our distinct pleasure to welcome you to our campus. We hope the coming days will bring stimulating dialogue, collegial connections, and moments of experiencing hospitality that reflect our commitment to this discipline we all serve. Welcome, and enjoy the conference.

Radana Dvorak Ph.D.
Saint Martin's University
CCSC-NW 2025 Conference Chair

2025 CCSC Northwestern Conference Steering Committee

Radana Dvorak, Conference Chair ..... Saint Martin's University
Richard Weiss, Site Chair ..... The Evergreen State College
David Pouliot, Program Chair ..... Southern Oregon University
Lucas Cordova, Papers Chair ..... Willamette University
John Stratton, Panels & Tutorials Chair ..... Whitman College
Peter Drake, Partners Chair ..... Lewis & Clark College
Richard Weiss, Speakers Chair ..... The Evergreen State College
Fred Agbo, Student Posters Chair ..... Willamette University

Regional Board - 2025 CCSC Northwestern Region

Ben Tribelhorn, Regional Representative ..... University of Portland
Bryan Fischer, Registrar ..... Gonzaga University
Lucas Cordova, Treasurer ..... Willamette University
Sharon Tuttle, Editor ..... Cal Poly Humboldt University
Lucas Cordova, Past Conference Chair ..... Willamette University
Radana Dvorak, Conference Chair ..... St. Martin's University
Lucas Cordova, Website Administrator ..... Willamette University

LLMs and the Shifting Landscape of CS Education*

Keynote

Ameeta Agrawal
Director of the NLP Lab at Portland State University

We are at a turning point: from personalized tutoring systems to automated code generation, AI is transforming the educational landscape in profound ways. For instructors, this shift raises urgent questions: How do we adapt teaching methods when students have unprecedented access to AI tools? How do we foster creativity, critical thinking, and problem-solving in an era of instant answers? This keynote invites participants to reflect on the evolving relationship between human and machine intelligence, and the implications for shaping the next generation of computer scientists.

Defining and Evaluating Faculty Service at Different Institutions*

Panel Discussion

Shereen Khoja[1], Lucas Cordova[2], Tammy VanDeGrift[3]
[1]Mathematics, Computer Science, and Data Science
Pacific University, Forest Grove, OR 97116
shereen@pacificu.edu
[2]Computer Science, School of Computing & Information Sciences
Willamette University, Salem, OR 97301
lpcordova@willamette.edu
[3]Computer Science, Shiley School of Engineering
University of Portland, Portland, OR 97203
vandegri@up.edu

1 Abstract

Universities typically divide faculty workload into Teaching, Research or Scholarship, and Service. However, what qualifies as service can vary from one institution to another[1]. This panel will explore the ways that service is conceptualized and integrated into faculty workload policies. Faculty members from three different universities will share how their institutions define service, set expectations, balance workloads, and strive for equitable service distribution among faculty. They will share how service is tracked, whether and how it is compensated or rewarded, and reflect on challenges that arise especially when it comes to advising loads, disparities in service, and balancing service and scholarship.

Members of the audience will be invited to share insights from their own institutions, and we hope the panel sparks a larger conversation about the role

of service in academic life. If you've ever wondered how others handle service, or are rethinking it at your own institution, this session is for you.

The panelists will address the following topics:

- **Definition of Service:**
  - How does your institution define service?
  - What activities count as service?
  - Is there a formal distinction between service to the department, institution, profession, and community?
  - Why is service necessary for the institution to run?
- **Categorization of Service:**
  - How is service accounted for in the faculty workload model (percentage of FTE, workload credits)?
  - Is service load tracked? If so, how?
  - Are there different expectations for pre-tenure, post-tenure, or clinical/instructor faculty?
  - How is service documented and factored into reviews, promotion, and tenure?
  - Are certain kinds of service more valued than others?
- **Equity and Distribution of Service:**
  - What mechanisms exist to ensure that service is equitably distributed across faculty?
  - How is invisible service or labor acknowledged?
  - What service category is most under-recognized at your institution?
  - How are service assignments made? Is it voluntary, assigned, or a mix?
- **Moving forward:**
  - What is one best practice regarding faculty service at your institution?
  - What is an example of one area where your institution could improve regarding faculty service?

References

[1] American Council on Education. Equity-minded faculty workloads: What we can and should do now, 2022. `https://www.acenet.edu/Documents/Equity-Minded-Faculty-Workloads.pdf` retrieved September 24, 2025.

The Many Futures of CS Education*

Panel Discussion

Julie Medero[1], Adam Blank[2], Lauren Bricker[3],
John Stratton[4], Zachary Dodds[1],
[1]Harvey Mudd College; Claremont, CA 91711
[2]California Institute of Technology; Pasadena, CA 91125
[3]University of Washington; Seattle, WA 98195
[4]Whitman College; Walla Walla, WA 99362
medero@cs.hmc.edu, blank@caltech.edu, bricker@cs.washington.edu
strattja@whitman.edu,dodds@g.hmc.edu

1 Summary

Across the U.S. and the world, the day-to-day practice of computational work is changing – neither as rapidly as some assert, nor as slowly as many of our students would prefer. For instance, AI use is now a professional expectation along some paths, and it's unclear whether that number should - or will - increase or decrease. AI, however, is simply 2025's most-hyped computational trend. Less visibly, changes are underway across professional expectations of computing skills. Across nearly every human endeavor and community, expectations for computational understanding and effectiveness are no longer novel - they are simply presumed, which dramatically disadvantages those without computing background. This rise of unspoken norms for computing savvy could well impact our students and our educational goals far more deeply than the next foundation model. This panel convenes five CS educators currently piloting pedagogical, curricular, and/or programmatic changes in response.

2 Student-directed pedagogy: Prof. Julie Medero, HMC

Coding artifacts, graded for correctness and style, have traditionally been the primary source of formative assessment in CS1 courses. In fall 2024's CS1, Prof. Julie Medero replaced artifact-based assessment with starkly restructured CS1 assessments, reweighting student deliverables away from autograded unit-tests and toward the agency and responsibility of in-person presentation of those artifacts. Now, deliverables are short, low-stakes "share-outs" during which students demonstrate and discuss the software systems they authored that week. This transformation has been both dramatic and successful, and it has prompted adoption across all of CS1. Prof. Medero will offer lessons learned, opportunities gained, and considerations for future improvements.

3 In-person assessment at scale: Prof. Adam Blank, Caltech

Caltech's spring 2025 offering of CS1 augmented submitted artifacts with in-person meetings named *DUE*s. Each DUE consists of a student Demo, showing a conceptual Understanding, and a novel Experiment. DUEs are a small portion of an assignment's points (15%), but they are required. Through our experience at Caltech, the mechanics of hosting DUEs have evolved so that, now, (a) every course instructor and TA helps host DUE meetings; (b) students sign up for a 30 minute window, at which multiple course staff are on duty, with instructor supervision; (c) each DUE involves multiple TAs, rather than a single, one-on-one interaction. Prof. Adam Blank will share how these adaptations successfully retain advantages of demonstration-based deliverables, even as they scale to a larger set of students.

4 Critical Inclusivity: Prof. Lauren Bricker, UW

At the University of Washington (UW), the number of Engineering degrees conferred from 2015-2024 rose by 44%, and the number of Computer science degrees rose over 168%. However, traditionally, only 25% of low income or first-generation students students as well as those from less well served high schools in Washington state have successfully earned a engineering or computer science degrees. With holistic support, however, the student success rate for these populations rises to 78% or higher. In 2013, the University of Washington College of Engineering began a two-year State Academic Redshirt Student (STARS) program focusing on engineering students. Support was added in the Computer Science and Engineering pathway in 2016. In 2017, the Paul G. Allen School at UW added a summer bridge program; in 2022, it was expanded

to a full one-year support program (Allen Scholars) in 2022. To date STARS has supported 390 students and Allen Scholars has supported 156 students.

Professor Lauren Bricker, who has been instrumental in developing summer bridge and supplemental courses for students in these programs, will discuss her position on the core support we can continue to provide in upcoming ten years in order to increase the retention and graduation of computer science and engineering students.

5 The CS Core Curriculum Exchanges Content for Personal Development: Prof. John Stratton, Whitman

One of the principles in the recent curriculum redesign at Whitman College was reducing the size of our "core" content in favor of greater "core" personal skill development. For example, our only required computer systems course will be CS 110: Computer Systems and Society, with no programming prerequisites or attempts to teach hardware design skills. In exchange, the course has room to explicitly develop skills such as looking up reference documentation for the computing tools they need to use, because the rest of the content will always be there for them if we train them to find and take in that knowledge for themselves when they need it.

6 Curricular changes: Prof. Zachary Dodds, HMC

In 2035, specifying processes from scratch is no longer IntroCS's core. Core instead is growing new cognitive models - our own, and possibly others' - for expressing, executing, and assessing processes. Which is to say, *We've leveled up!* In 2035, IntroCS mixes hands-on, how-it-works insight across far more computational models, e.g., string-passing via APIs, voltage-passing via wires, meaning-passing via embeddings, and value-passing via `return`, among others. Together, our students - and ourselves - are *seeing and celebrating* the water in which we all swim.

Acknowledgments

The authors gratefully acknowledge resources from NSF DUE #2142780, from Google, as well as support from our respective institutions.

Real-World Data, Interactive Games and Data Structure Visualizations in Early CS Courses Using BRIDGES*

Conference Tutorial

Kalpathi Subramanian Erik Saule
Department of Computer Science
The University of North Carolina at Charlotte
Charlotte, NC 28223, USA
{krs,esaule}@charlotte.edu

Grounding computer science concepts in real-world and socially relevant problems can be a key to increasing students' motivation and engagement in computing. The BRIDGES [1] infrastructure, developed at UNC Charlotte and in use for the past decade, facilitates easy access to *real-world data* and supports *visualization* capabilities for early CS courses. Since its inception, BRIDGES based tools have been adopted across more than 20 institutions and used by nearly 8000 students. Our evaluations show that using BRIDGES in data structures, algorithms, and other courses has resulted in improved student learning outcomes.

The BRIDGES API, with bindings for C++, Java and Python provides several key advantages important to early CS courses. First, minimal effort is needed to access real-world datasets; a single function call will return an array of objects corresponding to the dataset, for use by students as part of their assignment. Available data sets in BRIDGES span several domains, including entertainment (IMDB, actor-movie data, song lyrics, audio signals), science (USGIS Earthquakes), geographic data (Open Street Map, elevation maps, US and World maps), and literature (Project Gutenberg). Second, BRIDGES can be used by students to create and explore a visualization of the data and data structures they themselves implemented; such visualizations can be used to illustrate concepts of underlying algorithms and data structures, or highlight important features using its visual components. Third, the BRIDGES Game

API allows students to *implement 2D games* which can be used to emphasize and practice basic programming concepts (control structures, looping) and implement game logic, while providing a fun experience. Finally, students can benchmark algorithms in BRIDGES by using data sets that are both real and large. Benchmarking features help to illustrate algorithm performance (as a function of data size) and reinforce the importance of computational complexity. Multiple live examples of BRIDGES visualizations and example assignments can be seen on the BRIDGES website and in Fig. 1.

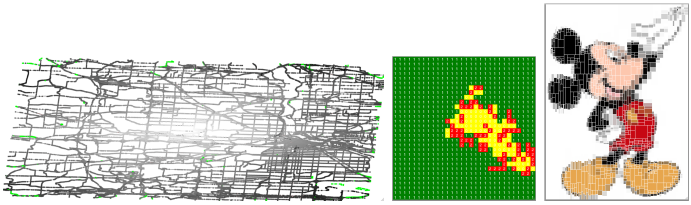


Figure 1: BRIDGES Examples. *[Left]* Dijkstra's shortest path algorithm applied to Minneapolis downtown using Open Street Map (here, gray values are lighter close to the source and darken with distance). *[Middle]* A simple fire spreading simulation exercise, *[Right]* An K-D tree representation of an image

Acknowledgements

The BRIDGES project has been funded by multiple grants from the National Science Foundation (Nos. 1245841, 1726809, 2142381)

Biography

Dr. Kalpathi Subramanian is an Associate Professor of Computer Science at the University of North Carolina at Charlotte. He is the PI on multiple NSF TUES and IUSE awards for the past 12 years that funded BRIDGES, along with Co-PIs Drs. Erik Saule and Jamie Payton.

Dr. Erik Saule is an Associate Professor of Computer Science at the University of North Carolina at Charlotte. His research interests include the education of parallel computing, data structures, and algorithms. He served as the Program Chair of the 2018 Workshop on Education for High-Performance Computing (EduHPC-18) and is co-PI of the NSF IUSE award.

References

[1] David Burlinson, Matthew Mcquaigue, Alec Goncharow, Kalpathi Subramanian, Erik Saule, Jamie Payton, and Paula Goolkasian. Bridges: Real world data, assignments and visualizations to engage and motivate cs majors. *Education and Information Technologies*, pages 1–27, 2023.

Supporting Hands-On Cybersecurity Exercises with A Human-in-the-Loop Hint Generation System*

Conference Tutorial

Taylor Wolff[1], Richard Weiss[1], Claire Simpson[1],
Joseph Granville[1], Jack Cook[1], Ishan Abraham[2],
Jens Mache[2]
[1]The Evergreen State College, Olympia, WA 98505
{taylor.wolff, weissr, claire.simpson}@evergreen.edu,
jwgranville@gmail, cookjackc@gmail.com
[2]Lewis & Clark College, Portland, OR 97219
{ishanabraham, jmache}@lclark.edu

As the role of AI in education continues to expand, an interesting question is: "What effects will AI have on the instructor's role in the classroom?" The technology shows potential in assisting both instructors and students of our cybersecurity education framework EDURange [3]. For example, previous work has explored how machine learning techniques such as classification could be used to predict a student's outcome for a given exercise [2, 1, 7, 6].

For generating and communicating hints that assist the student, AI dialog systems are a popular option; however, students may become overly dependent on them [5]. Thus, we developed EDUHints [4], a human-in-the-loop hint generation system for the EDURange platform that only instructors (including teaching assistants) have access to. This approach seeks to mitigate any over-dependence by students while preserving organic student-instructor interaction. In this interactive tutorial, we will demonstrate how an instructor (or a teaching assistant) can operate the system from their own UI dashboard. We will discuss many of the system's technical details, e.g, how it can generate hints entirely on cloud or local hardware via inference with llama.cpp and a small language model (SLM), and how we engineer prompts. We will also share our

progress with in-development features, such as a classifier for identifying which students may need assistance, and the implementation of more sophisticated methods for retrieval-augmented generation (RAG), such as GraphRAG.

Acknowledgements

This work was partially supported by the National Science Foundation under Awards 2216485 and 2216492.

References

[1] Aubrey Birdwell, Jack Cook, Richard Weiss, and Jens Mache. From logs to learning: Applying machine learning to instructor intervention in cybersecurity exercises. In *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference*, 2024.

[2] Quinn Vinlove, Jens Mache, and Richard Weiss. Predicting student success in cybersecurity exercises with a support vector classifier. *Journal of Computing Sciences in Colleges*, 36(1), 2020.

[3] R. Weiss, J. Mache, and M. Locasto. The EDURange framework and a movie-themed exercise in network reconnaissance. In *Proceedings of USENIX Security: Advances in Security Education Workshop (ASE)*, 2017.

[4] Taylor Wolff, Richard Weiss, Jack Cook, Joseph Granville, and Jens Mache. EDUHints: A human-in-the-loop small language model hint generation system for cybersecurity education. In *Proceedings of the 24th European Conference on Cyber Warfare and Security (ECCWS)*, pages 897–903, 2025.

[5] Chunpeng Zhai, Santoso Wibowo, and Lily D Li. The effects of over-reliance on AI dialogue systems on students' cognitive abilities: a systematic review. *Smart Learning Environments*, 11(1):28, 2024.

[6] Valdemar Švábenský, Kristián Tkáčik, Aubrey Birdwell, Richard Weiss, Ryan S. Baker, Pavel Čeleda, Jan Vykopal, Jens Mache, and Ankur Chattopadhyay. Detecting unsuccessful students in cybersecurity exercises in two different learning environments. In *Proceedings of the IEEE Frontiers in Education Conference (FIE)*, 2024.

[7] Valdemar Švábenský, Richard Weiss, Jack Cook, Jan Vykopal, Pavel Čeleda, Jens Mache, Radoslav Chudovský, and Ankur Chattopadhyay. Evaluating two approaches to assessing student progress in cybersecurity exercises. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, 2022.

An Introduction to C++ OpenMP Parallel Programming*

Conference Tutorial

Xuguang Chen
Department of Computer Science
Saint Martin's University
Lacey, WA 98503-3200
xchen@stmartin.edu

Parallel computing executes tasks simultaneously across multiple processors and is critical for meeting computational demands in fields such as scientific simulation and deep learning [5][7]. It accelerates complex calculations in physics, chemistry, and climate modelling, and shortens neural network training on GPUs. In academia, it is increasingly included in computer science and engineering curricula to meet industry needs [2][1]. Introductory courses teach programming models for efficient algorithm design, while advanced courses address GPU programming with CUDA or distributed systems with Hadoop through practical projects. Many programs also integrate it into areas like data science and AI, highlighting its interdisciplinary role.

Parallel computing models are generally divided into message-passing and shared-memory. MPI (Message Passing Interface) [6] is a standardized API for distributed systems, enabling explicit message exchange between processes on separate nodes. It is highly scalable and widely used in scientific simulation and big-data processing. OpenMP (Open Multi-Processing) [4][3] is a shared-memory API for multi-core CPUs, using compiler directives in C, C++, Fortran, or Java to parallelize loops or tasks without explicit thread management. Due to its simplicity, OpenMP is common in university courses and applied in areas such as physics and image processing. MPI and OpenMP are complementary [8]: the former suits distributed systems, the latter shared-memory environments.

In this tutorial, C++ OpenMP for parallel programming is introduced, aiming at beginners who have had experience with a programming language

and are interested in parallel computing. It covers essential operations, software setup, and program development, equipping learners with practical skills to harness parallelism. In addition, as the teaching materials, this tutorial is also suitable for a selected topic of a CS1 or CS2 course in university computer science programs, aiding students in enhancing their C++ proficiency. By the end, learners will confidently use OpenMP to improve C++ program performance on multi-core systems.

Learning Objectives of the tutorial are summarized below.

- Comprehend the fundamentals of OpenMP and its role in shared-memory parallel programming.
- Learn to install and configure necessary software for OpenMP development.
- Master writing, compiling, and running C++ programs with OpenMP directives.
- Apply key OpenMP constructs to parallelize loops and tasks effectively.

The tutorial includes the following topics.

- **Software Installation:** Instructions are provided for how to install a compiler with OpenMP support, such as GCC (via MinGW-w64 on Windows or natively on Linux/macOS) or MSVC (via Visual Studio).
- **Editing, Compiling, and Running:** The tutorial shows learners through a simple OpenMP program (e.g., parallel "Hello World") using an editor like MS Visual Studio. Compilation commands (e.g., g++ -fopenmp main.cpp -o main) and execution steps are explained.
- **Basic OpenMP Operations:** The core OpenMP directives are introduced, such as #pragma omp parallel for creating parallel regions and #pragma omp for for distributing loop iterations across threads. Clauses like private, shared, and reduction are explained to manage data scoping and prevent race conditions. Scheduling options (static, dynamic, guided) are covered to optimize workload distribution.
- **Practical Examples:** Hands-on examples, like parallelizing a matrix multiplication loop, demonstrate real-world applications will be involved.

At the end of the tutorial, the learners can be provided the e-version of the lecture notes, code as examples, and other materials for self-study, if needed.

References

[1] MIT OpenCourseWare, Parallel Computing course materials. `https://ocw.mit.edu/courses/18-337j-parallel-computing-fall-2011/`.

[2] Stanford University, CS149: Parallel Computing course syllabus. `https://gfxcourses.stanford.edu/cs149/fall24`.

[3] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 5.0, 2018. `https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-0.pdf`.

[4] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[5] Maria T. Gonzalez, Paul K. Fischer, and Elena R. Martinez. Parallel computing for large-scale genomic data analysis: Scalability and performance. In *SC '22: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '22, pages 156–164. IEEE Press, 2022.

[6] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI (2nd ed.): portable parallel programming with the message-passing interface*. MIT Press, Cambridge, MA, USA, 1999.

[7] Sarah K. Lee, David M. Brooks, and James T. Anderson. Optimizing scientific simulations with distributed parallel computing frameworks. In *PPoPP '23: Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '23, pages 89–97. Association for Computing Machinery, 2023.

[8] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Education Group, 2003.

Simulating Ethernet's Exponential Backoff Considered Helpful *

Joseph P. Skudlarek[1] and Jens Mache[2]
Department of Computer Science
Lewis & Clark College
Portland, Oregon
[1] Jskud76@alumni.princeton.edu
[2] jmache@lclark.edu

Abstract

In this paper, we describe our experience with having students simulate Ethernet's congestion control algorithm in an undergraduate computer science course; we used the assignment to enhance understanding and appreciation of networking, programming, simulation, problem solving, statistics, and analysis of experiments. Somewhat wide-ranging, the assignment was designed as a broad overview using Ethernet as the carrier to introduce many related computer science topics, with an additional focus on using skills already acquired in CS I/II to solve a real-world problem. The assignment also served as a vehicle to provide individualized feedback to the students on their programming practice.

In addition to sketching the history of Ethernet, we described in detail the truncated binary exponential backoff congestion control algorithm, noted that it's difficult to get accurate analytic results for other than a simplified model, and that one could, and we would, simulate it to determine the expected value for time needed for the first successful packet transmission starting with simultaneous senders, parameterized by the number of concurrent senders and the maximum backoff size.

Keywords: CS education, CS I/II, networking, discrete simulation, distributed algorithms, computing history, analysis of experiments, applied statistics

1 Introduction: Context and Motivation

Lewis & Clark College offers CS-293, which has the following description:

> Introduction to computer networks and web development. Topics may include internet protocols, client-server computing, distributed applications, databases.

The course prerequisite is the foundational course sequence CS I and CS II, which covers an introduction to programming, data structures, algorithms, and algorithm analysis. In CS I, for example, students simulate the Birthday Paradox problem[8].

Ethernet is a ubiquitous networking protocol used for more than half a century, and embodies foundational computer science topics. Being proficient at programming is an essential skill for a well-rounded computer scientist. A great way to teach computational thinking and programming is to have the students think for themselves, to program, and then give them detailed constructive feedback on their programs.

We used a one-hour class period to describe Ethernet, including the exponential backoff congestion control algorithm, and to assign homework. We used a subsequent class three weeks later to explore and discuss different solution techniques, compare theory with practice, provide feedback on programming, and show how to apply statistics, including normal distributions and the Central Limit Theorem.

2 The Assignment

> **The purpose of computing is insight, not numbers.**
> – Richard Hamming –

The accurate performance of the Ethernet congestion control algorithm is difficult to compute analytically — simulation to the rescue!

2.1 Intended Benefits

- encourage the students to think critically, and gain confidence by applying what they already know to solve a real-world problem
- improve student programming prowess by providing practice and personalized feedback
- encourage independent problem solving and analysis with a somewhat open-ended assignment — the goals were clear, various approaches were presented, but justification of their approach was left to the student

- provide a springboard for the real-world issues of understanding how to set up experiments and apply analysis to assess the validity of the results, bringing industrial experience into the classroom
- demonstrate, with a concrete example, one of the best distributed algorithms without global control, Ethernet's congestion control mechanism
- provide an opportunity to discuss and compare theory with reality — in particular, conventional theory uses an illustrative (but overly) simplified model which does not match the algorithm and results seen in practice
- motivate applying simple statistics including standard deviation; justify the validity, and emphasize the critical distinction between an underlying [*non-Gaussian*] distribution and the resulting *Gaussian* distribution of the expected value

2.2 Student Deliverables

> **The purpose of programming is communication.**
> – Joseph P. Skudlarek –

Programming was an essential aspect of this assignment. The students were encouraged to communicate clearly, that correctness and style are important, in both the program and the writeup. The students were given the following guidelines:

- keep writeup line length <= 80 characters
- keep program line length <= 120 characters
- pick good names
- prefer short focused functions
- minimize *global* ***variables***
- use helpful comments — don't need many; do need some

The assignment was to create and submit 2 files:

1. a self-contained Python program to simulate Ethernet's congestion control algorithm which
   - computes the expected time needed for the first successful packet transmission, as described in the abstract, parameterized by
     - number of concurrent senders
     - maximum allowed backoff depth
     - sample size
   - implements a driver to run the simulator and produce results
   - extra credit: produces graphs of interesting results

2. an ASCII text file, with answers the following questions:
    - how did you test your simulator?
    - how do you know it's right? (with the explicit guidance that it's OK to know you don't know)
    - how did you set the sample size?
    - how did you use AI if at all? (AI generated code was disallowed)
    - what did you find interesting about the assignment or the results?
    - what else do you want to mention?

To simplify grading, to structure the assignment, and to help define success, we provided the following specifications.

Using Python3, define `ev_oneOK(k,xb)` which returns the expected value (not an individual trial) for the time taken to send the very first packet successfully with `k` simultaneous senders and maximum backoff `xb`.

Set `max_k=14` and `max_b=4`. Run your program for each pair of values (`k`, `xb`) for `k` in `range(max_k+1)` and `xb` in `[max_b-1, max_b]`. Do yourself a big favor, and debug using much smaller values of `max_k` and `max_b`.

The results should be **within 2%** of the true value. This is an important point – we're writing the simulator to determine the true value – and the certainty depends on the simulator fidelity and the *sample size*.

Students were provided with the initial lecture slides, and a 6 page handout describing the algorithm and the assignment in detail.

Note that students did *not* turn in the output of their program — we read and ran every program to assess accuracy, ensure a level playing field, and to provide individual feedback.

3 Initial Presentation and Handouts

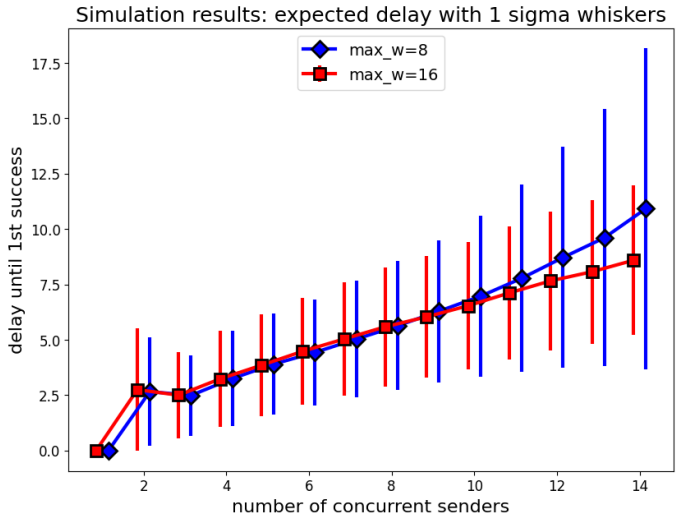
The initial presentation consisted of 16 slides, and emphasized

- The ISO 7 Layer Cake
- Ethernet History
- CSMA/CD and Exponential Backoff
- Hardware Signaling
- Error Detection and Correction

Figure 1 was especially helpful to illustrate the results sought, and show the trend lines.

The homework handout was 6 pages, gave a detailed description of the algorithm, and guidance on how to answer some questions, including how to use the standard deviation of the expected value to assess the estimated accuracy.

Figure 1: Delay until first successful transmission, with confidence intervals



3.1 Worked Example

We worked out a small example for how to structure the simulator in class on the board; here's one simple example we like, illustrated in Table 1 below; it shows a trial needing 3 time slots to send the first successful packet.

Consider 3 senders. Use a single array, `delay`, indexed by sender, to indicate the delay until attempted send; keep track of the current time in the variable `time`; and use a single array `window`, also indexed by sender, to track the size of sender k's backoff window.

At time `time=0`, all senders schedule to send with delay 1, with backoff window of size 1. Now repeatedly advance time until the first packet gets sent. To advance time, `time+=1`, and `delay-=1`. If there is is a single delay of 0, that sender sends successfully, so return `time`. If there is more than 1 delay of 0, those senders collide, so double their delay window `w` (up to a max), and reschedule them randomly in [1,`w`].

4 Establishing Ground Truth

To ensure that our grading was accurate, i.e., to assess if the student simulator-based predictions were within 2% of objective reality, we needed to establish the **ground truth**. So we carefully built and tested our *reference* simulator, parameterized for both initial and final backoff window sizes. We ensured it

Table 1: Illustration of a single trial's progress and outcome

time	delay[0]	delay[1]	delay[2]	window[0]	window[1]	window[2]
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	1	1	2	2	2	2
2	0	0	1	2	2	2
2	2	3	1	4	4	2
3	1	2	**0**	4	4	2

met ad hoc expectations for small test cases. And we compared it with the analytic solution of our *Detailed* model, described below.

The conventional theory [5, 7] uses a *simplified* model which assumes all backoffs have occurred and each future slot in the backoff window is equally likely. That does not match our situation — we simulate increasing backoffs, and just one slot in the backoff window is chosen for each sender — the conventional theory only roughly approximates our reality.

Even restricting the situation to rescheduling using only the optimal window size, the conventional model (*Simple*) provides overly-optimistic answers when compared with our more realistic model (*Detailed*).

Figure 2: Comparison of models

senders	*Simple*	*Detailed*
2	2.0	2.5
3	2.2500	2.3962
4	2.3704	2.5635

After satisfying the ad hoc validations, we compared our simulator output with the analytic solution of *Detailed*, our more accurate yet still tractable alternative model. It's the expected time to first successful send with k senders, all fully backed off, each of which has $2^{\texttt{xb}}$ random choices for their delays.

This model can be solved using a straightforward expected value calculation[2, 4]. E.g., for `k=4` senders and `xb=2`, *Detailed*'s solution has $(2^{\texttt{xb}})^k = 4^4 = 256$ states and 256 equations, computes the expected value for each state, then combines those individual values to produce the overall expected value. An example equation is `ev(0,1,2,3)=1`, where parameter$_i$ is the delay for sender$_i$ — in that state, there are no conflicts, and the first sender's packet is successfully transmitted, requiring 1 time slot. The matrix formulation can easily be created and solved numerically to full precision using Octave[3].

5 Scheduling and Grading

The 27 students had 2 full weeks to do this assignment, all extension requests were granted, and we had a week to grade it and provide detailed feedback;

several office hour sessions were offered each week over those next 3 weeks; only a few sessions were attended.

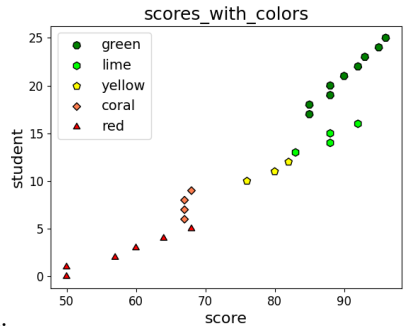
The submission rate was surprisingly *low* — more than half the students were late. With so many late submissions and the overriding desire to help the students learn by getting them to do the work and receive individual feedback, we extended the deadline through the weekend, which helped quite a bit.

The students were graded on both the writeup and the program. While grades are important to assess progress, our main focus was to promote learning and provide good feedback — and in some cases, great submissions got lengthy feedback.

Initially, we used a green/lime/yellow/coral/red grading rubric — a kind of incidental grading, in which we tried to focus attention on the learning and feedback, and not the letter grade[1].

We then made another review pass and mapped the resulting assessment into a numeric value, inspired by our preferred grading rubric[6]. Numeric values were needed for incorporation into the existing class mechanisms. Knowing the distribution was not Gaussian (the sample size was too small and had clustered values), we didn't grade on a curve. Grades were then assigned based on numeric values; the distribution was somewhat bimodal.

Figure 3: Grades with color-coding, students ordered by color & score



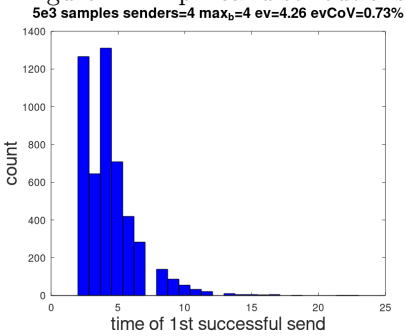
6 Wrap-up Presentation

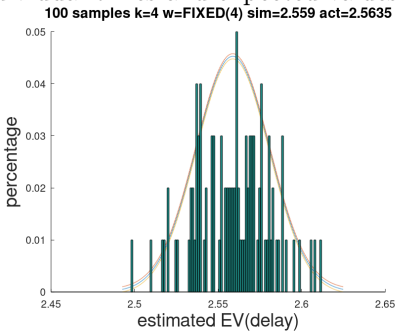
The wrap-up presentation consisted of about 30 slides, with heavy emphasis on graphs to show the effects of varying the sample size and the values of repeated runs. The board work included going back and forth between the underlying discrete distribution of individual times which retains the same lumpy shape as sample size increases, and the Gaussian distribution of expected values which gets narrower and smoother as the sample size increases; cf. Figure 4. [These graphs are for *different* experiments used to compare theory with simulation, and have different expected values.]

We addressed the these topics in the 1 hour follow-up class period.

- Central Limit Theorem
  We focused on explaining why we get a Gaussian Distribution for the computed expected value, to justify why it's OK to use the standard

Figure 4: Empirical distributions of individual times and expected values





deviation to estimate accuracy. We mentioned but didn't dwell on using the standard error of the sampling mean, preferring to give the more intuitive approach of running multiple estimates of the expected value, and using their standard deviation.

- Programming Practice and Pragmatics
  Selected suggestions based on the programming that was submitted. In particular, we reinforced good naming and modular decomposition.
- Python Idioms and Uglies
  There are many nice features, especially list comprehension, in Python, but also some ugly quirks (e.g., inconsistencies in interfaces) that can trip up unsuspecting programmers.
- Performance
  The focus of this exercise was clarity and accuracy, not performance. As we stated in the assignment:

  > Performance matters, but it's not the focus of this assignment. The program should run in under 1 minute — but could take up to 9X longer. Ensure that one can easily reduce `max_k` and `sample_size` to reduce the runtime.

  Most programs took under 10 seconds, and all programs ran in under 2 minutes.

7 Observations and Lessons Learned

Many students were unclear about how the algorithm worked despite a detailed prose description and examples in class — so we'll add a worked-out-in-detail example (like that shown above in Table 1) to both the initial presentation and homework handout to solidify and clarify the steps.

The initial presentation and homework handout should provide the specific numeric values for a few key values, including `senders=4` and `max_backoff=4`. We showed these values in class, but not everyone caught it.

One student said it would help to provide justification for the submission format — so we'll add that the submission handling was automated, and failure to follow the format forced annoying manual rework.

To make the homework assignment more free-standing (another student suggestion), we'll add Figure 1, which was in the initial presentation, to the homework handout.

There are a lot of moving pieces here; to reduce the load on the student, it would be better to consistently use just time-to-send (vs. delay-before-sending), and max-backoff (vs. max-window-size).

Some students found the distilled description of the algorithm in the homework too terse, and thought we failed to emphasize what's important. Adding the worked example will help.

One of our key observations was that some students tended to think that just because the program ran without errors and had a trend in the right direction, that was proof enough that it was correct; we know that's not true; now, hopefully they do too.

And many a student knew they didn't know for sure their simulator was correct, which is a major benefit — students experience they know when they know, know when they don't know, and that lack of error messages in the output does not imply correctness.

Other interesting student observations include the following:

- enjoyed seeing how earlier simulation of birthday paradox gave way to this simulation
- struggled with wrapping their head around the details, but when it clicked, it was very satisfying
- found it interesting to make predictions about how the data would behave and then compare that to the observed data [this student also successfully noticed and analytically indicated why there was the surprising increase in expected time needed at `k=2` vs. `k=3` senders]
- the first time I felt like I was applying CS I ideas in a class beyond CS I to understand new concepts ... we used Computer Science as a tool ... I enjoyed it, and I wish we had more of those

8 Conclusion

Reflecting on student submissions, grading, and feedback, we find this exercise successfully engaged and educated both the students and the instructors. Now

that the assignment is fleshed out and classroom tested, we recommend it to you.

For the current versions of materials used in the classroom, please contact the first author.

9 Acknowledgments

Lewis & Clark College, especially Profs. Peter Drake, Jens Mache, Alain Kägi, and Jeff Ely, as well as the students, provide a rich and rewarding environment for computer science education, enabling the 1st author to bring years of study and industrial experience into the classroom. In particular, Prof. Jens Mache (2nd author, taught CS-293) was encouraging and supportive as our initial lecture on Ethernet expanded into this two-part presentation with homework, grading, and feedback.

Thanks to the CCSC-NW anonymous reviewers for their beneficial feedback.

References

[1] Susan D. Blum, ed. *Ungrading: Why Rating Students Undermines Learning (and What to Do Instead)*. Morgantown: West Virginia U. Press, 2020.

[2] C. J. Cullen. Private Communication. Work directly with $\mathbf{ev}(state)$; {avoids discrete $\mathbf{pr}(k)$, recurrence relations, and $\Sigma\,(k \cdot \mathbf{pr}(k))$}. 2012.

[3] John W. Eaton et al. *GNU Octave version 7.1.0 manual: a high-level interactive language for numerical computations*. 2022. URL: `https://www.gnu.org/software/octave/doc/v7.1.0/`.

[4] William Feller. "An Introduction to Probability Theory and Its Applications". In: 3/e. Vol. 1. New York: Wiley, 1968, pp. 348–349.

[5] Robert M. Metcalfe and David R. Boggs. "Ethernet: distributed packet switching for local computer networks". In: *Commun. ACM* 19.7 (July 1976), pp. 395–404. ISSN: 0001-0782. DOI: `10.1145/360248.360253`. URL: `https://doi.org/10.1145/360248.360253`.

[6] Peter Drake. *Policies*. `https://github.com/PeterDrake/drakepedia/blob/master/administrivia/policies.md`. [accessed 04-July-2025].

[7] Andrew S. Tanenbaum. "Computer networks (3rd ed.)" In: Prentice-Hall, Inc., 1996, p. 283. ISBN: 0133499456.

[8] Wikipedia contributors. *Birthday problem — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Birthday_problem&oldid=1291691534`. [accessed 11-June-2025]. 2025.

Expanding Applications of Programming with Peer-Instructor Video Tutorials*

Richert Wang[1] and Kevin Buffardi[2]
[1]Computer Science
UC Santa Barbara
Santa Barbara, CA, 93106
richert@ucsb.edu
[2]Computer Science
California State University, Chico
Chico, CA, 95929
kbuffardi@csuchico.edu

Abstract

Broadening interest in learning how to program may require demonstrating its relevance in a greater variety of applications. We introduce an approach using "peer-instructor" undergraduate students to create short video tutorials on foundational concepts that represent a wider variety of contexts than traditionally found in programming courses. This paper shares strategies and challenges when facilitating peer-instructors' creation of unique and relevant video content. We also report qualitative analysis on the content of their videos, which indicates broader representation for applications of coding.

1 Introduction

Computer science (CS) education provides in-demand 21st century skills like programming (coding). In addition to its importance to traditional computing

careers, coding is transforming other disciplines while also spurring emerging fields, such as data science. However, only about half of US high schools even offer foundational courses in computing [7]. Meanwhile, university CS majors disproportionally consist of Asian (17.7%) and white (33.85%) men compared to the general population [1]. Consequently, there is a need to reach broader audiences and relate coding to a variety of different interests and applications.

Traditionally, coding classes heavily emphasize applications such as creating video games, digital electronics, and robotics. In previous work [4], we found that women were less likely to express interest in those applications. Our previous study also revealed that students identified many other hobbies and interests with broad appeal across race and gender barriers; however, students did not perceive how coding could be relevant to those interests.

Researchers and educators have explored different educational interventions to appeal to marginalized students such as women, Black, Indigenous, and people of color. However, we also recognize potential risks with prescriptive, top-down approaches. For example, if we featured marginalized students to project diversity without honoring their cultural capital and unique perspectives, it would demonstrate *tokenism of minorities* [2]. Similarly, *pinkwashing* [9] describes the practice of marketing to girls and women through superficial gestures like pink aesthetics without serving their genuine needs. Meanwhile, emphasizing stereotypical experiences and interests exemplifies the harmful practice of *minority essentialism* by assuming intrinsic experiences, behaviors, or dispositions to people based on their gender or racial demographic [13].

Instead, this paper describes our approach to empower students with the agency to identify and describe unique applications of coding *as it relates to their lives*. We investigated the research question: **Would affording peer-instructors the agency to create videos that are relevant to their own interests and experiences represent a wider variety of themes and applications of coding?** We recruited undergraduate students with unique perspectives and paths into learning how to code, and supervised their creation of tutorial videos that explain coding concepts in a variety of novel contexts.

In this paper, we provide an experience report on recruiting and mentoring peer-instructors in creating relatable CS educational videos derived from their unique interests and perspectives. We also analyze recurring themes our peer-instructors selected when presenting CS concepts in their own words and compare the breadth of those themes to those our previous research found to have broad appeal [4].

2 Background

In previous work [4], we surveyed students (who had not yet taken a collegiate-level CS courses) to list their interests; in the subsequent question, we asked whether they considered those interests relevant to computer science. In analyzing students' free-written responses (not limited in length nor content), we performed thematic analysis and identified a wide range of themes including traditional STEM (e.g. electronics and physical sciences); other academic disciplines; games; automobiles; fashion and cosmetics; arts and crafts; reading; music, film, and audio/visual; sleep and relaxation; socializing; athletics; nature and outdoors; culinary; travel; and animals. However, of students who self-identified each of those interests, a majority only perceived any relevance to computer science for traditional STEM, games, automobiles, and other academic disciplines.

In that study, we analyzed students' interests, as disaggregated by gender, race, and major (either CS or non-major). While we found no statistically significant differences by race or major, men were disproportionately more likely to express interest in games and traditional STEM hobbies than women and non-binary students [4]. Anecdotally, we observed that CS learning materials often reflect and reinforce students' misconceptions that coding is mainly relevant to applications like robotics and video games. Meanwhile, other interests like arts/crafts, music/film/audio/visuals, and athletics showed broad appeal across demographics, and could be emphasized more to provide more relatable coding examples. Accordingly, we posited that empowering students to create unique coding examples that are relevant to *their* interests will represent a greater variety of learning materials that are more relatable to students who have interests other than just video games and electronics.

Education research has shown that learning from fellow students—such as through near-peer mentoring and peer instruction—provides multiple benefits to students. Peer-instruction (PI) is an in-class active learning method that prompts students to apply a concept and then explain it to peers while trying to resolve differences in their understanding [8]. Meanwhile, studies of near-peer mentors—mentors who are slightly ahead of their protégés in their academic careers—have shown that students' interest in computer science can be improved with access to role-models that are relatable and resonate with their identities; relatability is a significant predictor of self-efficacy and interest in computer science [6, 11]. Near-peer computer science mentoring has also improved computing confidence and identity for the near-peers, and students improved learning, competency, and interest in computing [12].

Our work focuses on mentoring (near) peer-instructors to promote a sense of belonging, relatability, and role models to students learning how to program in secondary school or introductory college courses. Our peer-instructors share

their insights asynchronously via video recordings that may not have the same benefit from some of the one-on-one interactions of PI and near-peer mentors, however the videos are scalable and accessible to students who may not have direct access to mentors, role-models, and relatable content. Students that do not have a direct shared experience with peer-instructors can still form an appreciation of their own and others' cultures, and broaden their understanding of various applications of CS not immediately apparent in their lived experiences.

3 Method

We hired undergraduate students as *peer-instructors* to record short programming tutorials in various introductory programming content with Java, C++, and Python languages. The concepts for Java align with the Advanced Placement Computer Science A (AP CSA) modules[1]. We identified concepts common among introductory C++ programming at UC Santa Barbara and CSU Chico including data types; variables; boolean operators; decisions; arrays; vectors; functions; pass-by-reference and pass-by-value parameters; for loops; while loops; structs; classes; recursion; and pointers. The C++ videos covered comparable material as they align with most CS1 university courses by adjusting for language-specific distinctions (e.g. C-style arrays and vectors instead of Java ArrayLists). Similarly, we identified introductory Python concepts including data types; variables; IO; Strings; math operations; boolean operators; decisions; lists, dictionaries; tuples; for loops; while loops; file IO; functions; sets; and pandas[2] data analysis functions such as reading from a CSV file.

3.1 Recruiting Peer-Instructors

To recruit peer-instructors, we advertised an open call for applications from undergraduate students at UC Santa Barbara and CSU Chico. The call for peer-instructors was not limited to CS-related majors, and any undergraduate student capable of presenting introductory CS concepts was considered. In the recruiting process, we asked applicants to specify their programming experience (including courses completed), their pathway to taking programming courses, and an explanation or demonstration of how they could teach an introductory CS topic of their choosing in a unique context. While demonstrating some fundamental programming aptitude was required, our recruitment prioritized applicants' communication skills and abilities to explain coding in unique contexts.

[1] https://apstudents.collegeboard.org/courses/ap-computer-science-a
[2] https://pandas.pydata.org/

The peer-instructors selected for this project demonstrated clear communication skills, an approachable demeanor on camera, a unique and inspiring pathway into CS, and a passion for helping students. In the initial phase of our research, we recruited six students to record C++ videos. In a second phase, we hired three from UC Santa Barbara for Java videos and three from CSU Chico for Python videos. In total, we have hired twelve peer-instructors (n=12). UC Santa Barbara and CSU Chico are Hispanic-Serving Institutions and even though the recruitment and selection of peer-instructors had no basis in immutable demographic characteristics, the peer-instructors hired included a diverse mix of race/ethnic and gender representation as well as a variety of academic majors, including Computer Science, Computer Engineering, Philosophy, Mechatronic Engineering, and Mechanical Engineering. At the time of their roles as peer-instructors, their academic level also ranged from freshman (first year collegiate) to senior (four-or-more years, near to graduation). Either by coincidence or by the emphasis on unique perspectives and/or pathways to coding, every peer-instructor reflected at least one marginalized demographic.

3.2 Creating Tutorial Videos

In order to support peer-instructors maintaining a steady throughput of completed video tutorials, each supervisor followed a weekly cadence with their team to discuss what topic(s) each peer-instructor was working on, brainstorm and discuss peer-instructors' examples and how it related to the programming topic, and provide feedback on any video that was a work in-progress. We reviewed and curated videos to ensure the topics were described accurately with appropriate code. When necessary, peer-instructors made new recordings to address issues with their work-in-progress videos. Anecdotally, peer-instructors also learned from each other during weekly meetings to make best use of the recording studios and video-editing software, share presentation tips, and generate novel applications from each other.

In previous work [5], we wanted recordings to highlight peer-instructors' personalities / identities using a Lightboard [10] to capture drawings, gestures, and their facial expressions. Since then, we improved the studio environment by overlaying a code editor with the Lightboard allowing peer-instructors to live-code their solutions. Details on this studio setup are described in [3] and a screenshot of a video using a Lightboard with the Visual Studio Code IDE overlay is shown in **Figure 1**.

We purposefully encouraged the peer-instructors to come up with their own unique examples relevant to their lives instead of dictating what examples students should present. Our general guidance to the peer-instructors concentrated on creating concise videos (about 5-10 minutes) that walk through an example related to their unique perspective of a programming topic. We

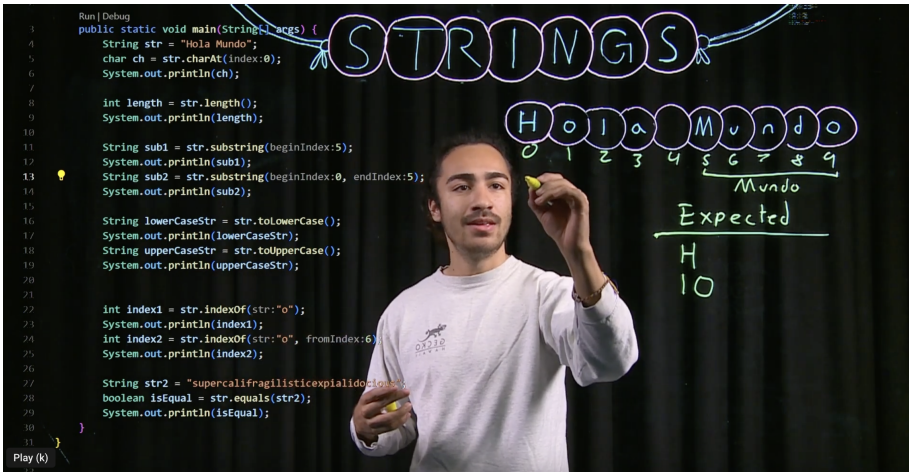


Figure 1: Example of overlaying Visual Studio Code Editor on the Lightboard to explain how Java Strings work using word bracelets.

encouraged them to code simple examples that can fit entirely on the screen and be live-coded, enabling a learner to code along with their example. When recording the videos, we directed the peer-instructors to explicitly explain why their example was relevant to their life. To prime peer-instructors to think of code in contexts relevant to their lives, we first assigned each one to create a video introducing themselves and stating how / why they pursued CS. We have also published these "Why We Code" videos[3] so that students can get to know the peer-instructor, and potentially identify with or feel motivated by them.

4 Results

At the time of preparing this manuscript, our peer-instructors recorded 94 videos. To investigate the common themes in the variety of applications demonstrated by our peer-instructors, we performed thematic analysis. Six of the videos were introductory videos (as explained in **Section 3**) that did not demonstrate specific coding concepts nor concentrate on singular themes. We also identified two early videos (before we adopted the introductory videos as part of the mentoring process) where peer-instructors explained a concept, but did not explicitly connect the concept with their lives. After excluding those videos, we performed thematic analysis to identify which theme was represented in each video.

[3]https://youtube.com/@codewit/videos

Independently, the two authors of this paper coded the theme that describes the main application demonstrated in each of the videos, based on the themes identified in our previous research on students' interests [4]. The initial coding had approximately 74% agreement and a moderately strong Cohen's Kappa measurement of inter-rater reliability ($\kappa = 0.73$). Of the videos with different themes identified, fifteen of the twenty (74%) were videos where one or both of the authors identified that its theme did not match any of the pre-defined themes (as enumerated in **Section 2** and shown in **Table 1**).

We discussed each item of disagreement and converged on the primary theme of each video (100% agreement, $\kappa = 1.0$) and established additional themes to represent those that did not match any from previous research [4]. The new themes include **adult responsibilities** (e.g. managing budgets, searching for housing), **productivity** (e.g. taking notes, converting computing file formats), **spirituality and belief** (e.g. horoscopes), and **toys and collectibles** (e.g. *Lego* constructions, collecting *Funko* figurines). **Table 1** also identifies themes where our previous research [4] revealed statistical *disproportionate interest*. Specifically, men were more likely to express interest in *traditional STEM* (36%) and *games* (68%) in comparison to women and non-binary students (14% and 36%, respectively).

Table 1: Themes in Peer-Instructor Videos, Relative to Students' Interests

Theme	Interest	Perceived Relevant	Disproportionate Interest	Peer-Instructor Videos
Traditional STEM	31%	98%	Men	8%
Games	60%	83%	Men	6%
Other Academic Disciplines	10%	70%	-	1%
Automotive	10%	50%	-	2%
Fashion and Cosmetics	3%	40%	-	8%
Arts and Crafts	30%	38%	-	7%
Reading	15%	30%	-	2%
Music, Film, and Audio/Visual	34%	29%	-	12%
Sleep and Relaxing	4%	29%	-	0%
Socializing	13%	15%	-	2%
Athletics	54%	12%	-	16%
Nature and Outdoors	25%	12%	-	4%
Culinary	7%	7%	-	7%
Travel	4%	0%	-	1%
Animals	2%	0%	-	1%
New Themes				
Adult Responsibilities	-	-	-	**14%**
Productivity	-	-	-	**4%**
Spirituality and Belief	-	-	-	**1%**
Toys and Collectibles	-	-	-	**2%**

Table 1 shows the peer-instructor videos represented a breadth of different applications of CS. The most popular themes were **Athletics** (16%), **Adult Responsibilities** (14%), and **Music, Film, and Audio/Visual** (12%). Each percentage is rounded to the nearest integer so totals do not add exactly to 100%. It is worth noting that even without direct instruction to represent specific themes and respecting peer-instructors' agency, every theme

was represented at least once, besides *Sleep and Relaxing*. While our previous research found that most students *did not* perceive CS as relevant to most of their interests, **83% of the peer-instructor videos demonstrated applications of coding that students previously saw no connection to CS** [4]. Consequently, we answered our research question and observed that *by affording peer-instructors the agency to create videos that are relevant to their own interests and experiences, the peer-instructor videos represent a wider variety of themes and applications of coding without additional interventions.*

5 Discussion

Our work demonstrates preliminary evidence that by enabling peer-instructors to create videos based on their own interests, we can create more diverse CS materials represented both by the peer-instructors and the coding applications they related to a variety of lived experiences. However, our study does not include a control (such as comparison to existing textbooks or other YouTube videos), which poses a **threat to validity**.

Future work is necessary to compare our results to outcomes in a controlled study. Currently, we have targeted introductory-level computer science programming topics in popular languages, but there is an opportunity to further expand video instruction to cover more topics that may resonate with a larger audience including, but not limited to, scientific computing, game development, digital art, and data science.

We purposefully gave peer-instructors free agency on explaining topics that were relevant to their lives. Giving this freedom was important to us in order to avoid essentialism and have peer-instructors express topics that resonated with them the most. This resulted in a broad range of themes, but we noticed that peer-instructors did not particularly focus on their personal culture and traditions. Even though all perspectives promote peer-instructors' unique cultural views, future work can include more targeted mentoring to promote cultural themes relevant to peer-instructors in order to broaden these perspectives to further support culturally responsive pedagogy.

6 Conclusion

This paper details our experiences recruiting and mentoring undergraduate students as they create unique coding examples that relate to their lived experiences. In our project, we have noted a broader representation of programming applications than what is typically perceived as relevant programming contexts. In particular, out of the 86 videos analyzed, the top themes represented were

Athletics, Adult Responsibilities, and Music/Film/Audio/Visual; these themes represent more variety than students' perceptions of coding applications [4].

In the approach shared in this work, peer-instructors were purposefully given freedom to speak about CS in their own ideas, analogies, and applications. We expect the process and materials shared in this paper are beneficial to peer-instructors as well as future learners who consume their worked example videos.

All the materials described in this paper are available at `https://codewit.us` and on YouTube[3] as open educational resources so that any teacher or student interested in unique perspectives and applications of coding can benefit from the diverse and broad representation. Access to these videos can asynchronously scale to all students interested in learning programming and also be used as supplemental learning in existing curriculum, with the benefit of connecting with peer-instructors and developing an appreciation of how programming can be applied in various contexts. This should be beneficial to classrooms that lack diversity and representation in their current curricula, but may be particularly advantageous to students who lack access to computing courses in their primary/secondary education.

7 Acknowledgments

This material is based upon work supported in part by the National Science Foundation (award #2315883) and by the Learning Lab, an initiative of California Governor's Office of Planning and Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Learning Lab.

References

[1] Computing Research Association. *Statistics & Data Hub, Postsecondary Computing Degrees Awarded (Version 1.3.7). Center for Evaluating the Research Pipeline*. 2021. URL: `https://bpcnet.org/statistics` (visited on 02/09/2025).

[2] Johanna Blumenthal et al. "Diversity and its role in computing resources for further reflection: computing for the social good in education members". In: *SIGCAS Computers in Society* 49.2 (Jan. 2021), pp. 15–16. ISSN: 0095-2737. DOI: `10.1145/3447903.3447911`. URL: `https://doi.org/10.1145/3447903.3447911`.

[3] Kevin Buffardi. "CodeVid Studio: Coding Videos with Multimodal Instruction". In: *Journal of Computing Sciences in Colleges* 38.10 (Apr. 2023), pp. 26–34. ISSN: 1937-4771.

[4] Kevin Buffardi and Subhed Chavan. "Is programming relevant to CS1 students' interests?" In: *Journal of Computing Sciences in Colleges* 37.1 (Oct. 2021), pp. 45–53. ISSN: 1937-4771.

[5] Kevin Buffardi, Elena Harris, and Richert Wang. "Codewit.us: A Platform for Diverse Perspectives in Coding". In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. SIGCSE 2022. Providence, RI, USA: Association for Computing Machinery, 2022, pp. 780–786. ISBN: 9781450390705. DOI: 10.1145/3478431.3499398. URL: https://doi.org/10.1145/3478431.3499398.

[6] Jody Clarke-Midura et al. "How Near Peer Mentoring Affects Middle School Mentees". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 664–669. ISBN: 9781450351034. DOI: 10.1145/3159450.3159525. URL: https://doi.org/10.1145/3159450.3159525.

[7] Advocacy Coalition. *Advocacy Coalition - State of CS*. 2024. URL: https://advocacy.code.org/stateofcs/ (visited on 02/09/2025).

[8] Catherine H Crouch and Eric Mazur. "Peer instruction: Ten years of experience and results". In: *American journal of physics* 69.9 (2001), pp. 970–977.

[9] Amy Lubitow and Mia Davis. "Pastel Injustice: The Corporate Use of Pinkwashing for Profit". In: *Environmental Justice* 4.2 (2011), pp. 139–144. DOI: 10.1089/env.2010.0026. eprint: https://doi.org/10.1089/env.2010.0026. URL: https://doi.org/10.1089/env.2010.0026.

[10] M. Lubrick, George Zhou, and Jingsheng Zhang. "Is the Future Bright? The Potential of Lightboard Videos for Student Achievement and Engagement in Learning". In: *Eurasia Journal of Mathematics, Science and Technology Education* 15 (Apr. 2019). DOI: 10.29333/ejmste/108437.

[11] Oluwakemi Ola. "Using Near-Peer Interviews to Support English Language Learners". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 952–958. ISBN: 9781450394314. DOI: 10.1145/3545945.3569868. URL: https://doi.org/10.1145/3545945.3569868.

[12] Jennifer Rosales et al. “The Experience of Near-Peer Computing Mentors: Strengthening and Expanding Women’s Computing Identities in Undergraduate Interdisciplinary Contexts”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 1154–1160. ISBN: 9798400704239. DOI: 10.1145/3626252.3630946. URL: https://doi.org/10.1145/3626252.3630946.

[13] Michelle Trim. “Essentialism is the enemy of the good: how the myth of objectivity is holding computing back”. In: *SIGCAS Computers in Society* 49.2 (Jan. 2021), pp. 11–13. ISSN: 0095-2737. DOI: 10.1145/3447903.3447908. URL: https://doi.org/10.1145/3447903.3447908.

Computing History Case Study: The Japanese Fifth Generation Computing Systems Project – A Reassessment *

Kayako Yamakoshi[1,2], J. Paul Myers, Jr.[1]
[1]Department of Computer Science
Trinity University
San Antonio, TX 78212
pmyers@trinity.edu
[2]Senior Data Analyst, Uber
kayamakos@gmail.com

Abstract

The curriculum often neglects computing's history. An interesting study involves the Japanese Fifth Generation Project that startled the world in the early 1980s. Unlike previous hardware generations, this was primarily software-based (AI with logic programming). After fifteen years, the Project could be seen as a technical failure. But a full reassessment also includes other explicit goals: international participation, research environments, human resource development, reinvigoration of AI, and creating an enduring Japanese computing industry.

1 Introduction

Within epsilon, nothing gets remembered - nobody remembers history.
– Edward Feigenbaum, Turing Award Recipient, on computing's past [4]

The lack of interest, the disdain for history is what makes computing not quite a field. – Alan Kay, Turing Award Recipient [11]

Sadly, computing history is largely neglected in the CS curriculum. As with ethics, awareness of this history should be encouraged [16]. A particularly interesting topic is the Japanese Fifth Generation Computer Systems Project (FGCS, 5Gen), a decade-long period of forty years ago. Investigating this Project is especially rewarding because it is relatively self-contained and it generated a highly emotional response in the U.S. and the West. Incorporating 5Gen into a brief teaching module will perforce touch on applications, hardware, software, and theory - all relevant for AI research. Also relevant are issues of national identity, cultural biases, and worldwide cooperation.

This is the last of a trilogy of papers, constituting a small curricular module: background and overview [17] and proposals for incorporating 5Gen in the curriculum [16]. This third is a reassessment dealing with earlier misconceptions. Earlier assessments focused on technical aspects of 5Gen, leading to a sense of failure of the Project. The present focuses on more intangible aspects, but aspects that were nonetheless explicitly part of the overall goals. From this new perspective, we see that in many respects 5Gen was a success.

FGCS was an extensive national Project 1982–1992, follow-on through 1994; final reporting concluded in March 1995. Created by the Institute for New Generation Computer Technology (ICOT) and headed by its research director, Kazuhiro Fuchi, ICOT consisted of researchers from the Ministry of International Trade and Industry's (MITI) Agency of Industrial Science and Technology. It also included participants from several companies: Oki, Sharp, Toshiba, Nippon Electric, Hitachi, Fujitsu, Matsushita Electric, and Mitsubishi Electric [15]; and universities such as Tokyo, Kyoto, Tohoku, and Tsukuba [10]. FGCS was designed to foster an unprecedented collaboration among government, industry, and academia, a unified goal that fueled anxieties in the West [17].

In 1979, MITI directed JIPDEC (then Japan Information Processing Development Corporation; now Japan Institute for Promotion of Digital Economy and Community) to create the FGCS Research and Development Committee for the Fifth Generation computers. The Committee also hosted the International Conference on FGCS to gather the opinions from experts worldwide [10].

A goal was a Knowledge Information Processing System (KIPS) with advanced inference capabilities, exceeding the capabilities of already robust expert systems. Regarding standard databases, Fuchi had observed: "At present it is common that databases and programming languages belong to different systems. This is not a desirable situation. Their unification appears to be quite feasible" [7]. Indeed, as has long been asserted in the logic-programming (LP) community, the program = the data! "The logic program serving to define the data serves at the same time to compute it" [2]. Thus, 5Gen was primarily software-based: AI, using a new LP language: Prolog, where the program and

the database are the same system.

Although ICOT prototypes were not intended for sale, the vision targeted eventual broad, mass use, relying on special parallel hardware, natural language capability, and miniaturization. The "surprisingly specific" specifications included a high-performance parallel Prolog machine, bit-mapped display, natural human-computer interface (HCI), compact size, and "beautiful appearance." There were no plans to sell the machines; they would be Project research tools [23]. Fuchi focused on making a machine to meet the needs of people [8]: users would communicate intelligently with the computer using natural language and images. Fuchi emphasized natural language, image interaction, and AI; aspects of this vision resonate with today's mobile devices and AI assistants. Additional FGCS details are available elsewhere [9, 18].

Three phases were planned for this very well-funded Project:

> [T]he initial stage is envisioned that software and hardware modules are built and ... experimental systems configured by integrating these modules ... [to] include hardware and software simulators, prototypes for language processing, and experimental natural language processing systems. The intermediate stage is ... [for] integrating them into inference and knowledge-base subsystems. In ... the final stage, ... [t]he total system is developed, integrating the subsystems in order to define the ultimate goals clearly [9].

The presumption that government, industry, and academia would be able to work harmoniously caused considerable anxiety in the West. In their 1983 book, McCorduck and Feigenbaum wrote: "We are writing this book because we are worried," reflecting the West's alarm over the FGCS Project [5]; but nothing quite captures the anxiety better than the CACM's September 1983 visual (visceral) cover, also made into a poster [3, 16, 17]. So, the West was alarmed; seemingly well-founded since Japan already dominated the automotive and consumer electronics industries. The U.S. founded an umbrella corporation, MCC (Microelectronics and Computer Technology Corporation), to respond to the perceived threat. Other countries developed responses, as well [18].

Much of the anxiety was due to a stereotype that Japanese "national character" (that the individual is second to the group/society) would lead to harmonious coordination among the three sectors. However, FGCS was plagued by secrecy and competitiveness - as was predicted, for example, for the U.S. attempt. Matsumoto argues and cites additional studies indicating that there is no support for the claim that Japanese are particularly collectivistic [13, 20]. Especially in competitive academic and profit-making corporate environments, the stereotype fails. Anyway, as discussed below, the goal of FGCS was not to achieve worldwide dominance in computing; rather there were several goals oriented toward building technical capabilities within Japan itself.

In any event, the decade ended and the Project disappeared with no closure and with seemingly few appreciable results; FGCS was deemed by many to be a failure. Given the initial concerns and reactions of the West, it is odd that not much has been written on the outcomes of the Project; it has essentially dropped out of historical awareness in the computing field ... even in Japan [17]. The primary focus had been on the technological and that can most easily be regarded, in part, as a Project failure. But many have misunderstood the full FGCS purposes. So, the objective here is to reconfirm the goals, technical and non-technical, of FGCS and to note accomplishments in technologies, research environments, international contributions, and personnel development. That the emphasis solely on technological issues is misdirected is apparent from early statements by ICOT and Fuchi:

> *Autonomy from the immediate concerns of the computer industry* not only freed the FGCS research agenda from the constraints of profit-seeking, it allowed ICOT to commit itself to the scientific norms of openness and disinterestedness. Fuchi signaled this commitment by regularly and painstakingly emphasizing that the Project had *"no commercial objectives."* The lack of corporate influence was, of course, unusual for a massive computing project, Japanese or otherwise. MITI, for example, was running several other major computing projects concurrently with the FGCS, all of which required participating firms to commercialize their results. But the flagship *FGCS Project was to demonstrate Japan's new leadership role in science and technology.* ... Because "Our project is an effort to create a new age for modern man," explained Fuchi, it *"must not be exclusive or closed to outsiders."* Foreign researchers would be invited to participate, and results published in English as well as Japanese, including at conferences abroad [8] (italics ours).

Further, Warren speculates as to why the aims of the Project were revealed so openly; noting that priorities were for prestige, pure research on a large scale, and joining the international research community [23]. Indeed, the very fact of many international conferences exploring 5Gen implies that proprietary technology was not the main focus.

2 Outcomes of Stated Goals of the FGCS Project

In addition to hardware and software, goals included reinvigorating Japan's computing industry through enhanced international participation, research, and human resources. Each will briefly be examined for this reassessment.

2.1 Technology: Hardware and Software

There were technical innovations in parallel processing and non-von Neumann computers that likely influenced the developing era of parallel/distributed computing, knowledge-base, LP, and inference systems leading to KIPS for medicine and law [9]. ICOT was also working on natural language as a way to apply the KIPS [9]. But most of the worldwide attention to FGCS was concentrated on its goals of intelligent, personal, and small-scale HCI, now largely regarded as the points of "failure." However, Feigenbaum and Shrobe, in an extensive evaluation of 5Gen technology, came to regard these assessments of failure as "unrealistically negative" [6, 12]. Because that 1993 assessment is comprehensive and detailed, these matters will not be discussed further here.

Japan was often characterized as hardware-strong but software-weaker; FGCS explicitly targeted software and inference capabilities, and so a goal of FGCS was to remedy this. They realized that a new software paradigm would be important in dealing with inference in a large KIPS. However, more recently Feigenbaum has noted that the even newer paradigm of machine learning would soon obviate the widespread need for a KIPS and its LP inference capability [4]. It is worth noting that dramatic increases in computer speed led to the use of standard programming languages for AI, further decreasing the perceived need for specialized machines and languages (including LISP). So, interest in Prolog and the complexities of its unification/resolution methods waned [21]. For a discussion of elementary logic programming, see Myers [16].

Note on "AI winter." It is often acknowledged that AI has experienced several "winters." Given the keen interest of the West in FGCS, it seems that 5Gen was the kickstart needed to end the 1980's "winter" with an AI "spring." Feigenbaum, however, challenges this simple, tidy, and convenient conclusion [4]. An AI "winter" had persisted into the early 1980's, and "[AI] research would not revive on a large scale until 1983, when Alvey (a research project of the British Government) began to fund AI again ... in response to the Japanese Fifth Generation Project," there was also a "winter" in the early 1990's [24]. Consistent with this latter lull, Feigenbaum added that "winters" are not unique to AI; all scientific initiatives have periods of intense work followed by quiet periods as new knowledge is assimilated to prepare for the next surge of activity [4]. Moreover, he notes that the notion of a winter is something of a chimera; for example, "soft computing" (neural networks as opposed to formal logic rule "hard computing" systems) was being developed in Japan as an underground phenomenon while all the funding initially went to FGCS. However, if for no other reason than the fear of Japanese domination, FGCS must be viewed as a factor in stimulating AI research worldwide.

Note on Expert Systems. FGCS was not a project on expert systems. There were already many usable expert systems by the early 1980's [5]; the

5Gen knowledge systems are much more general. Not just a database of facts, a KIPS adds to facts: "prejudices, beliefs, and, perhaps most important, heuristic knowledge ... the means to make sensible judgments." As an aside, this well predates the functionality of contemporary LLMs.

2.2 Research Environment Enhancement

Japan was lagging: "In Japan, little effort has been made in research on the key technologies, particularly software and basic theories" [9]. As noted above, the FGCS was not for strengthening Japanese industries, per se, but for conducting fundamental research. The environment for the FGCS was superb and well-funded: "not only was this the largest budget for any of MITI's computing projects, the FGCS was the first such project to run for a full decade" [8]. According to Norio Murakami, former President of Google Japan, only three expensive DEC SYSTEM-20's existed in Japan before the FGCS started; by its end, nine companies had installed these systems to participate in the Project. [Murakami was involved in FGCS at the Project's start as a minicomputer system engineer at DEC Japan, 1978 to 1985. He was responsible for installing a SYSTEM-20]. So, ICOT had built infrastructure for researchers. The Research Institute for Advanced Information Technology was established within JIPDEC with about twenty projects being launched. Also, Parallel Inference Machines (PIM's) were moved to several Japanese research institutions for further research. The FGCS Project had created a robust, sustained computing and research community and hardware infrastructure for Japan [14]. In terms of initiating and furthering Japanese research in computing, FGCS succeeded.

2.3 International Participation

From its outset, FGCS cooperated with similar projects in different countries. Additional projects inspired (or triggered) by FGCS are DARPA's Strategic Computing Program (and its revealing report, "Strategic Computing and Survivability") in the U.S. Others included the European Strategic Program for Research and Development in Information Technologies, the Sweden Institute of Computer Science, Argonne National Laboratory, the National Institute of Health, Lawrence Berkeley National Laboratory, the Australian National University, and the University of Bristol [22]. In addition, ICOT budgeted for inviting researchers from abroad to visit; researchers at ICOT were also encouraged to communicate with peers worldwide [18]. It is worth noting how unlikely these initiatives would have been if the primary objective of FGCS had been domination of the technology industry, as so feared by many. "There is clear evidence that the Japanese planners viewed their Project as having more than technological implications. They saw it as a positive step in the creation

of a new international relationship for Japan with the West, indeed with the whole world" [12]. In fact, early on, non-Japanese companies were invited to become partners in the enterprise; while none opted in, several Japanese subsidiaries became "general supporting members," including Burroughs, DEC, Canon, Sanyo, IBM, Xerox, Hewlett-Packard, etc. [18].

Japan is now regarded as a major player on the world computing stage. Indeed, when it was announced (May 2022) that the U.S. computer, Frontier, was then the fastest in the world, the computer it overtook was the Fugaku in Japan [1]. Regarding this and, for example, advanced robotics, fuzzy computing systems, and gaming that Japan produces, FGCS succeeded in helping to jump-start Japan's ascendance in the computing world internationally.

2.4 Human Resource Development

Fuchi stated that the major purpose of FGCS was to develop human resources. Much planning was intended to secure a sufficient budget [9]; ICOT's outstanding environment was instrumental in recruiting researchers. For example, it was difficult for a single company to purchase a DEC SYSTEM-20; thus, the well-equipped ICOT facilities were appealing to young researchers. Murakami notes that ICOT hired young researchers from companies and universities for each phase. In part, this was accomplished by evaluating AI undergraduate theses. Fuchi insisted that there be an age limit (30) for new researchers because developing young researchers was a major goal of the Project. Also, he felt that the typical seniority regimen in Japan would impede advances by those younger who were more knowledgeable about contemporary computing [14]. Hence, his employment practices were disruptive to the old establishment.

ICOT encouraged researchers to participate in international conferences as well as collaborative projects. This was also unique as the research budget for business trips was very limited in Japan. ICOT also budgeted to invite prominent scholars, whereby young researchers could attend lectures to broaden horizons [10]. FGCS researchers were encouraged to visit organizations abroad. As many scholars and organizations worldwide were interested in the Project, they were happy to welcome ICOT researchers. Some researchers returned to their home companies after finishing their terms at ICOT; others became leaders of research labs. At least thirty researchers became professors at prestigious universities in Japan, teaching, and disseminating the knowledge gained throughout the Project [14].

ICOT established working groups consisting of scholars not employed by ICOT [10]; there was a scholarship program for researchers from abroad to join FGCS. ICOT hired hundreds of young researchers, valuing performance over seniority. The researchers were rotated every three years so that more young workers in the field would get exposed to various environments. Representative

researchers were also sent to universities to educate upcoming students about FGCS, so that researchers from different institutes, public organizations, and private companies mingled and worked together [14]. Furthermore, the Project influenced the public; newspapers and magazines wrote articles, and those furthered the public's interest in computer technologies.

So, an outcome of the Fifth Generation Project is that "it trained hundreds, perhaps thousands, of engineers in advanced computer science. It is this training, rather than any particular piece of software, that will be the Project's legacy, as the engineers proceed to apply their skills at their respective Japanese companies" [19]. Hence, the training and placement of hundreds of Japanese (mostly young researchers) in advanced Computer Science professions is certainly a legacy and a genuine FGCS success.

3 Summary

Early on, Fuchi declared that this is the time for a "bold proposal" to develop "knowledge information processors with a nontraditional architecture and presenting a more natural HCI for the user" [7]. He later followed up: "Ten years ago we faced criticism of being too reckless. Now we see criticism from inside and outside the country because we have failed to achieve such grand goals" [8]. Outsiders, he said, initially exaggerated the aims of the Project, with the result that the program now seems to have fallen short of its goals [19].

However, the Japanese FGCS Project accomplished several goals in technologies, international participation, research environment enhancements, and technical human resource development. These accomplishments fulfilled stated goals of the Project and cannot be overestimated. Typically, in Japan, most students do not stay in academia, but work for companies after graduating. As a result, there was a shortage of professors who could teach the next generation in Computer Science. Thanks to FGCS, more qualified people went to universities, enabling a promising future.

By the formal end of the FGCS in March 1995, technologies, international networks, research environments, and human resources developed during the Project continued influencing the Computer Science world in many ways. For instance, AI and natural language processing are now commonplace in our daily lives. Is it too much of a stretch to regard the FGCS Project as possibly having influenced the eventual design and technologies of modern smart phones and their attendant AI "assistants"? At the final report meeting, Shunichi Uchida said, "The Fifth Generation Computer Systems Project was a project for the dandelion to bloom, and the follow up project was a project for the dandelion to spread its seeds" [9]. To some extent those seeds, that were spread by 1994, are in full bloom now and are spreading seeds for the next generations.

4 Acknowledgments

Gratitude is due to too many to list. But we must thank Dr. Edward Feigenbaum and Mr. Norio Murakami for their time and insights. Also, the staffs at JIPDEC, University of Tokyo CS Library, and National Diet Library for providing access and assistance. Dr. Yoshihiro Omura, hosted us at Kindai University, Summers 2016-2019; and Drs. Mark Brodl and Mario Gonzalez for continual encouragement. Also, ChatGPT-5 for fixing my many LaTeX errors! The authors were funded by a Trinity University Summer Undergraduate Research Fellowship and a Faculty Summer Research Stipend, respectively.

References

[1] D. Clark. "U.S. Retakes Top Spot in Supercomputer Race". In: *The New York Times* (May 30, 2022).

[2] V. Dahl. "On Database Systems Development Through Logic". In: *ACM Transactions on Database Systems* 7.1 (1982).

[3] D. Dittrich. "Cover Image". In: *Communications of the ACM* 26.9 (Sept. 1983).

[4] E. Feigenbaum. *Recorded phone interview with Myers*. May 2017.

[5] E. Feigenbaum and P. McCorduck. *The Fifth Generation: AI and Japan's Computer Challenge to the World*. Reading, MA: Addison-Wesley, 1983.

[6] E. Feigenbaum and H. Shrobe. "The Japanese National Fifth Generation Project: Introduction, Survey, and Evaluation". In: *Future Generation Computer Systems* 9 (1993).

[7] K. Fuchi. "Aiming for Knowledge Information Processing Systems". In: *Proceedings of the International Conference on Fifth Generation Computer Systems*. Oct. 1981.

[8] K. Fuchi. "Launching the New Era". In: *Communications of the ACM* 36.3 (1993).

[9] ICOT. *At the End of the Development of the Fifth Generation Computer Systems: Analects of Kazuhiro Fuchi and Shunichi Uchida from ICOT Journals*. Tokyo, 1995.

[10] ICOT. *Outline of Research and Development Plans for Fifth Generation Computer Systems*. Tokyo, 1983.

[11] A. Kay. *Interview with Alan Kay by A. Binstock*. 2012.

[12] K. Koizumi. “Technology at a Crossroads: The Fifth Generation Computer Project in Japan”. In: *Historical Studies in the Physical and Biological Sciences* 37.2 (Mar. 2007).

[13] D. Matsumoto. *The New Japan: Debunking Seven Cultural Stereotypes*. Yarmouth Maine: Intercultural Press, 2002.

[14] N. Murakami. *Interview with Yamakoshi*. 2017.

[15] N. Murakami. *Introduction to Knowledge Base Systems – Easy Artificial Intelligence*. Tokyo: Information Science Co., 2014.

[16] J.P. Myers. “The Japanese Fifth Generation Computing Project: Curricular Applications”. In: *CCSC-SE Conference and Journal of the CCSC* (Jan. 2021).

[17] J.P. Myers and K. Yamakoshi. “The Japanese Fifth Generation Computing Project: A Brief Overview”. In: *CCSC-RM Conference and Journal of the CCSC* (Oct. 2020).

[18] Y. Nakamura and M. Shibuya. *Japan’s Technology Policy: A Case Study of the Research and Development of the Fifth Generation Computer Systems*. Tokyo: Research Institute of International Trade and Industry, MITI, 1995.

[19] JIPDEC (Japan Institute for Promotion of Digital Economy and Community). *Research Report of the Fifth Generation Computer Systems*. Tokyo Seibunsha, 1981.

[20] Y. Takano and E. Osaka. “Japanese Collectivism and American Individualism: Re-examining the Dominant View”. In: *Shinrigaku Kenkyu: The Japanese Journal of Psychology* 68.4 (1997).

[21] M. Triska. *The Power of Prolog*. 2022. URL: `https://www.metalevel.at/prolog` (visited on 08/12/2025).

[22] S. Uchida. “General Report of the FGCS Follow-on Project”. In: *Proceedings of FGCS’94*. ICOT. Dec. 1994.

[23] D. H. D. Warren. “A View of the Fifth Generation and its Impact”. In: *AI Magazine* 3.4 (1982).

[24] Wikipedia. URL: `https://en.wikipedia.org/wiki/AI_winter`.

Using Lambda Calculus to Develop Educational Tools*

Abel Kelbessa[1] and David G. Wonnacott[2]
[1]Haverford College '25
abel.urgessa@gmail.com

[2]Haverford College CS Department
Haverford, Pa 19041
davew@cs.haverford.edu

Abstract

Some instructors have found substitution-based execution models helpful for introducing challenging concepts such as recursion at the introductory level. Tools such as the DrRacket Stepper and N-Dolphin help students understand this model; in some cases, this approach requires variable-renaming to preserve correctness, i.e., avoid the *name-capture problem*, raising the question of how to present this step to beginners; this presentation must be both *correct* and *clear*.

Church's λ-calculus is a fundamental tool for reasoning about substitution-based execution. Alpha(α)-conversion is the process of renaming variables in λ-calculus. We have found this framework provides a valuable tool for understanding correctness; discussions with, and a survey of, our CS1 students have guided our thinking about clarity.

Here, we discuss when and how α-conversion should be used to provide correctness or clarity in program-execution visualizers for CS1 students. In addition, we have identified specific misconceptions, for a preliminary Concept Inventory for renaming during α-conversion.

1 Introduction

Lambda(λ)-calculus [2, 3] provides a minimal foundation for performing computation in terms of *substitution*, just as "nand gates connected with wires" provides a minimal foundation for the physical realization of computation. Substitution is particularly helpful for understanding pure-functional code, and is often explored in courses or tools that introduce functional programming [6, 4]. Notional machines [1, 12, 13] based on substitution are (unsurprisingly) referred to as *substitution notional machines*. This framework provides novel ways to approach challenging CS1 programming topics such as recursion [14]. Substitution also underlies algebraic proofs about code and algorithms, such as those seen in/after Discrete Mathematics, and can provide insight when reasoning informally.

By providing a minimal formulation in terms of mathematics, λ-calculus provides a framework for (relatively) concise proofs about substitution-based tools. The α-conversion and β-reduction steps of un-typed λ-calculus are fundamental to our study of substitution-based computation, and will be the focus of much of this paper; topics such as β-abstraction, η-conversion, the Y combinator, currying, Church Booleans, and Church numbers will be touched on only briefly, to retain focus on the tools we're studying.

After recently demonstrating that our N-Dolphin software [10] was *not* sound, we became interested in reformulating its logic in a way that is both semantically sound and pedagogically appropriate. This paper documents our efforts to do so via λ-calculus. We contribute a more concise description of *how to use α-conversion during β-reduction* than we have so far found in the literature, alongside our *variant of this rule* that our students prefer for clarity, in Section 4.3. Additionally, we summarize some of what we have learned about pedagogical challenges as a preliminary *concept inventory for naming and renaming*, in Section 4.1.

2 Background on Lambda-Calculus

As mentioned earlier, λ-calculus enables us to solve complex problems in a pure functional approach. All algorithms and data are represented in terms of functions taking a single argument, e.g., mapping an input variable (parameter) x to an output expression M as:

$$x \to M$$

To represent the above in formal λ-calculus, we use the following terms:

- a "dot" or "." to separate the input, x, from the output, M
- Use the λ symbol to bind the input x

The above results in a function:

$$\lambda x.M$$

This approach of representing functions is called λ-abstraction. β-reduction is the process of simplifying a λ expression given an input [9]. In pure λ-calculus, each function takes and returns other functions, but techniques such as Church Booleans and Church Numerals can be used to encode boolean and integer values, so we will freely mix symbols like 1 into our examples to make our descriptions more accessible to those not fluent in λ-calculus. We will also use binary functions and operations like +, which would be handled by a technique called *currying* [9] in pure λ-calculus, and convert from the prefix notation that's traditionally in λ-calculus into the more common infix notation where that seems more natural, e.g., writing $4 + 1$ rather than $+1\ 4$ below.

Let's take a simple example where we have an integer x that we want to add 1 to, effectively incrementing it by 1. The function application would look like as follows in λ-calculus:

$$\lambda x. + x\ 1$$

or, in infix notation,

$$\lambda x.x + 1$$

To give an input such as 4 to the above function, we write

$$(\lambda x. + x\ 1)\ 4$$

and then us β-reduction, i.e., replace all x's after the "." with 4, giving

$$+\ 4\ 1 \text{ or, in infix notation } 4 + 1$$

The use of Church numbers for 4 and 1, and the analogous definition of +, lets us reduce this to the Church-number representation of 5, though these details are not important here.

The more formal expression for the above would be:

$$(\lambda x. + x\ 1)\ 4 \rightarrow_\beta 4 + 1$$

with β representing beta reduction. Having hinted at what β-reduction is thus far, let's define it. Beta-reduction is the process of transforming λ expression using substitution or function application, resulting in a simplification of the λ-expression [9]. As we have already observed, the λ-expression $(\lambda x. +\ x\ 1)$ was transformed by the application or substitution of the value 4 into it, but note that the parameter (4, here) and function body ($+\ x\ 1$, here) can be arbitrary expressions, including an expression that use one or more variables not declared within the code we're reducing (known as "free variables"). This raises a problem known as *name capture*.

The name capture problem [9] becomes evident when we attempt β-reduction on the following:

$$((\lambda x.\ (\lambda y. + x\ y))\ y)$$

The y that's used in the expression $+\ x\ y$ is bound to the inner λy. The second y, on the far right side of our expression, is not bound (i.e. free) in the original expression above. But, when we substitute it for x in the expression above, we're left with $\lambda y.+\ \ y\ y$. The *free* variable y has been "captured" by the parameter y, changing the meaning of the expression.

To avoid the name capture problem, we can change the name y in the inner λ-abstraction to avoid the free occurrence of the outer y from conflicting, as follows:

$$((\lambda x.\ (\lambda y.+\ x\ y))\ y) \rightarrow_\alpha ((\lambda x.\ (\lambda z.+\ x\ z))\ y) \rightarrow_\beta (\lambda z.+\ y\ z)$$

This approach of renaming variables is called α-conversion. In the α-converted version of the function, $((\lambda x.\ (\lambda z.+\ x\ z))$, we see that the bound variable y has been renamed from y to z, averting the name capture issue. The result here is a function that takes any number (z) and adds it to whatever y was; the earlier version produced a function that takes any number and doubles it, which is not at all the same thing.

3 Tools and Naming

The main challenge that we wanted to tackle in this paper is how we can understand multiple variables with the same name. If a student does not have the proper understanding of scoping and shadowing, they can easily get confused in understanding recursion or function calls. While it is possible to avoid variables with the same name in the context of original code, shadowing is something that is often inevitable in recursion.

It is often the case that students, when learning programming, start off with imperative programming ideas such as direct memory manipulation, mutable state, explicit side effects, sequential and parallel execution. On the other hand, pure functional programming offers features such as pure functions, referential transparency, high-order functions, and function composition. We teach a multi-paradigm CS1 [5, 16, 15] with strong support for visualization of notional machines [1, 12, 13]. We use Python Tutor [7] and a standard debugger for the usual dictionary notional machine; and N-Dolphin [10] and refactoring in an IDE such as PyCharm to visualize the substitution notional machine [14].

As mentioned earlier, when variables have the same name in the original code, we can rename before starting execution (either in the tool, or by hand when creating the example). However, conflicts can also arise during execution, e.g., of recursive code. Our tools respond to this challenge in different ways, which we'll illustrate using the non-recursive example in Figure 1, in which a call can bring together multiple variables with the same name.

Figure 2 shows the execution of the code in Figure 1 in the Python Tutor [7] tool. For easier viewing, we manually shortened some names, chang-

```
x: int = 25
def func_use_x(i: int) -> int:
    return i+x            # Use closest x ... but, what's closest?
def create_another_scope_call_func_use_x():
    x: int = 17           # another x, scoped inside "create..."
    return func_use_x(x)  # This uses the x that has value 17

print(x)
print(create_another_scope_call_func_use_x()) # what does it print?
print(x) # both printings of x out here show 25
```

Figure 1: Code with Multiple Variables with the Same Name

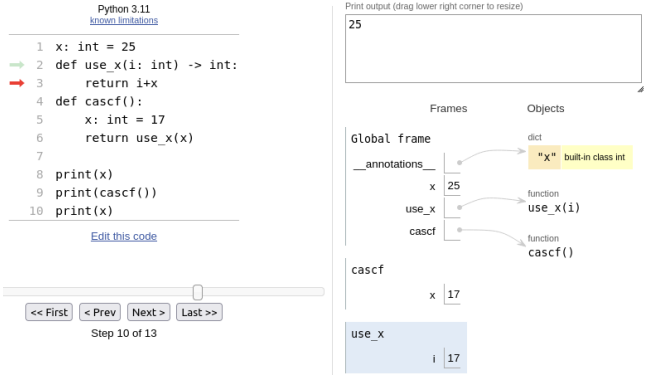
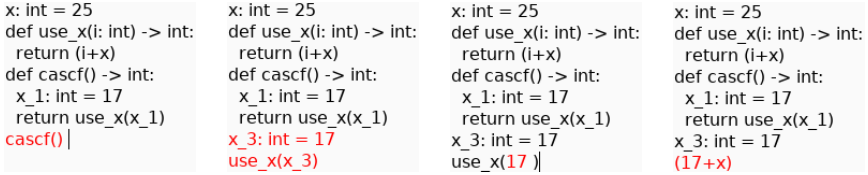


Figure 2: Python Tutor Visualization of the Code in Figure 1

ing `create_another_scope_call_func_use_x` to `cascf` and `func_use_x` to `use_x`. To figure out what will be returned for `i+x` on Line 3, the student must find the values of the *correct* variables in the "Frames" column (as must our software tools). There is only one `i`, but two `x`'s, and the choice of `x` depends on how the functions are organized, not just the text `i+x`.

Figure 3 shows four steps of the substition-based execution of this same code in the latest version of N-Dolphin, and illustrates several of the automatic-renaming features. Even before we've done any actual execution (Figure 3a), the `x` inside `cascf` has been automatically renamed `x_1` to avoid confusion with `x`. Execution of the call to `cascf` corresponds to substituting its body for the call (Figure 3b). During this process, a fresh `x` is created for the variable copied from `x_1`, and given any fresh name beginning with `x` (`x_3`, here). Only freshly-created variables are renamed, so the `x` of `i+x` remains unchanged when `use_x(17)` is executed (Figure 3c to 3d), and the result is unambiguous.



(a) starting code (b) inline `cascf` (c) inline `x_3` (d) inline `use_x`

Figure 3: N-Dolphin 1.0.α1 Visualization Execution of Figure 1

```
def square(n: int) -> int:
  also_n: int = n
  return (also_n*also_n)
((square(5)-11)*(52-square((3+4))))
```

a) Survey Fig. 2, original code

```
also_n: int = 5
n = 3 + 4
n1: int = n
print(((also_n * also_n) - 11) * (52 - (n1 * n1)))
```

b) Fig. 5, inlining with PyCharm

```
def square(n: int) -> int:
  also_n: int = n
  return (also_n*also_n)
((( lambda
      also_n=5:
    (also_n*also_n) )()-11)*(52-( lambda
     also_n=(3+4):
    (also_n*also_n) )() ))
```

c) Fig. 3, expr. inlining w/N-Dolphin

Figure 4: Simpler Code Used in Student Survey (using N-Dolphin 0.11)

Of course, the result would have been unambiguous with *either* of those automatic renamings, we didn't have to do both. This raises the questions "*How much renaming is helpful for beginners?*" and "*How much, if any, renaming is necessary to produce the correct result in a substitution-based system?*". The latter is discussed in the next section, but, for now, note that we don't get the right answer in Figure 3 without at least one of those renaming steps.

To begin to answer the question of how much renaming is *helpful*, we conducted a small student survey in Haverford College's CS1 course, using code from Figure 4a. For this code, unlike that of Figure 1-3, renaming is not required if we keep the definitions of `also_n` local to the expressions in which they're used (via clever use of `lambda` in Python). Before this survey, students had seen normal uses of `lambda` in Python, as well as automatic renaming during function inlining with the PyCharm IDE, as in Figure 4b (Fig. 5 in their survey), and had briefly been shown that `lambda` can be used to put a variable definition within an expression, as in Figure 4c (Fig. 3 in their survey).

The survey asked students their preference among three options:

1. Doing what N-Dolphin does now, i.e., Figure 3 [4c] shows two different variables that are both named `also_n`, in two different sets of parentheses, in addition to the third (original) `also_n` up in the function body.

2. Having a slight variant of Figure 3, in which the `also_n`'s in the parenthesis would be named `also_n_1` and `also_n_2`, providing the clarity of the actual structure, which we see in N-Dolphin, with less potential for confusing two variables with the same name.

3. Doing different things for simple or complicated cases, along the lines of what PyCharm does in Figure 5 [4b].

The result was an overwhelming support for Option 2, with eight students picking it. One student chose Option 1, and no student picked Option 3. This response, while from a small test group, shows that there is support for *consistent* renaming whenever we'd otherwise show two variables with the same name. This seems appropriate to us unless the CS1 course emphasizes subtleties of scoping rules and shadowing; we are content to point out when a fresh variable is created (though automatic renaming), and leave subtleties for a later course. Note that the DrRacket stepper [4] follows a similar approach to renaming, consistently adding numeric subscripts, though this is required in that context because variables are moved to the outermost (global) level when the function is inlined; PyCharm inlining seems to rename selectively.

4 Rules and Concept Inventory for Naming/Renaming

A concept inventory serves as a way to test a person's knowledge on a specific topic and see whether their understanding fits accepted conceptions or is in accordance with common misconceptions [8]. In our research, we came up with some misconceptions about α-conversion, as seen below. We also managed to draw out fundamental rules for renaming and when it must/should happen.

4.1 Preliminary Alpha-conversion Concept Inventory

We have identified several misconceptions related to variable names and renaming. A full concept inventory should be discussed with faculty from a variety of institutions, and then validated; we offer this list as an initial starting point.

- *Misconception* 1: α-conversion, by itself, changes a function's behavior. This is not true. It is used to avoid the name capture problem during β-reduction and to provide clarity.

- *Misconception* 2: A bound variable can be renamed to any variable name without introducing an error. This is not correct, as renaming a bound variable is allowed if and only if the new variable name is fresh and does not clash with existing free variables. This is easily achieved by using fresh variable names for each α-conversion.

- *Misconception* 3: A free variable can become bound after substitution without creating trouble. Correction, a free variable must not become bound after substitution. Thus, the importance of variable freshness during renaming to preserve the same functionality of the function.
- *Misconception* 4: Bound variables can be the same as free variables. No, every bound variable must be distinct from free variables that are used in the scope of that bound variable.
- *Misconception* 5: A free variable can be renamed using α-conversion. This is not true; α-conversion should only be used on a bound variable, across its entire scope, to ensure consistent changes to all uses.

4.2 Rules for Renaming

The rules for renaming are described in most λ-calculus references [9]. Briefly,

1. A free variable must not become bound after substitution. For instance, let's say we have the following function:

   $$(\lambda p.B)A$$

   In other words, let B be the body of the function that we are trying to substitute a *free* variable A into. We can substitute A into B if and only if A does not become bound after substitution.

2. Renaming a bound variable is allowed if and only if the new variable does not clash with existing free variables.

3. Bound variables must always be distinct or different from free variables. Looking at the example under point (1), if the variable A becomes bound after substitution in $(\lambda p.B)A$, that means there is a bound variable that has the same name as A in B. This breaks rule (1) for renaming, i.e., we have a *name-capture* problem, and should rename the bound variable that is in B using α-conversion.

4.3 When and How Should α-Conversion Occur in CS1 Tools?

An educational tool should be both *correct* and *clear*; thus, there are two main scenarios for renaming in a substitution-based tool used in CS1:

- We *must* perform α-conversion to preserve correctness when substituting a variable that will become bound after substitution. To generalize on the last rule above, consider what happens when A is an arbitrary argument expression with free and bound variables, when β-reducing

$$(\lambda p.B)A$$

We must α-convert all names that appear free in A but are bound in B in a context where p is used. Those names must remain unchanged in A, but be α-converted where they are bound and used in B, replacing them each with a fresh name that is not free in A (a completely fresh name can, of course, be used).

- We *recommend* α-converting when it can bring clarity to our program's execution without obscuring an important point. In our CS1 survey, students showed strong preference for *consistent renaming*, though in an upper-level class with more discussion of language semantics, it may be appropriate to toggle between consistent and only-when-needed renaming (or, possibly, no renaming, to show an incorrect execution).

5 Conclusion

We see λ-calculus as a tool for reasoning about student-facing tools for substitution-based learning. In particular, λ-calculus provides a concise logic for reasoning about substitution, with many standard definitions and results. These results, along with pedagogical concerns, guide our tool-construction efforts.

We believe appropriate tools are vital for substitution-based introductory teaching, and share the hope that this approach will help students with challenging concepts such as recursion [14, 11]. We have also found that it can help students see the connection between mathematical concepts and computer science, in a code-centric context that may be more comfortable for those who might not have a strong mathematical background.

Our exploration of renaming, and formalization via λ-calculus, have improved the experience of our students and the design of N-Dolphin in several ways. While we do not present λ-calculus itself to our CS1 students, it has helped us to remove one bug already, and guides our ongoing work toward a soundness proof of a revised version of our substitution axioms [16].

Our discussions with students, and surveys, help us understand the aspects of variable-naming that are challenging for beginners; N-Dolphin continues to develop to reflect this understanding. The latest version always α-converts to avoid duplicate names, though in future releases we hope to have a button to toggle between this mode and "rename only when necessary", as well as an option to hoist variable declarations out of expressions (as the DrRacket stepper and PyCharm do).

We invite thoughts from other educators, regarding these choices and our preliminary concept inventory.

References

[1] Benedict Du Boulay. "Some Difficulties of Learning to Program". In: *Journal of Educational Computing Research* 2.1 (1986), pp. 57–73. URL: https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9.

[2] Alonzo Church. "A set of postulates for the foundation of logic". In: *Annals of Mathematics* 33.2 (Apr. 1932), pp. 346–366. DOI: 10.2307/1968337.

[3] Alonzo Church. "An unsolvable problem of elementary number theory". In: *American Journal of Mathematics* 58.3 (Apr. 1936), pp. 345–363. DOI: 10.2307/2371045.

[4] John Clements, Matthew Flatt, and Matthias Felleisen. "Modeling an Algebraic Stepper". In: *Proceedings of the 10th European Symposium on Programming Languages and Systems*. ESOP '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 320–334.

[5] John P. Dougherty and David G. Wonnacott. "Use and Assessment of a Rigorous Approach to CS1". In: *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '05. St. Louis, Missouri, USA: Association for Computing Machinery, 2005, pp. 251–255. ISBN: 1581139977.

[6] Matthias Felleisen et al. *How to design programs: an introduction to programming and computing*. MIT Press, 2018.

[7] Philip 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*. SIGCSE '13. Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 579–584.

[8] Sally Hamouda et al. "A basic recursion concept inventory". In: *Computer Science Education* 27.2 (2017), pp. 121–148. URL: https://doi.org/10.1080/08993408.2017.1414728.

[9] Kenneth C. Louden and Kenneth A. Lambert. *Programming Languages: Principles and Practice*. 3rd. Boston, MA: CENGAGE Learning, 2012. ISBN: 978-1-111-52941-3.

[10] Alex Reichard and David G. Wonnacott. "N-Dolphin: A Visualizer for Abstract Substitution-Based Execution". In: *J. Comput. Sci. Coll.* 39.3 (Oct. 2023), pp. 267–276. ISSN: 1937-4771. URL: http://cs.haverford.edu/projects/N-Dolphin.

[11] Gulesh Shukla and David G. Wonnacott. "On Teaching and Testing Recursive Programming". In: *J. Comput. Sci. Coll.* 38.3 (Nov. 2022), pp. 98–106. ISSN: 1937-4771.

[12] Juha Sorva. "Notional Machines and Introductory Programming Education". In: *ACM Trans. Comput. Educ.* 13.2 (July 2013). URL: https://doi.org/10.1145/2483710.2483713.

[13] Juha Sorva, Ville Karavirta, and Lauri Malmi. "A Review of Generic Program Visualization Systems for Introductory Programming Education". In: *ACM Trans. Comput. Educ.* 13.4 (Nov. 2013). URL: https://doi.org/10.1145/2490822.

[14] Preston Tunnell Wilson, Kathi Fisler, and Shriram Krishnamurthi. "Evaluating the Tracing of Recursion in the Substitution Notional Machine". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 1023–1028.

[15] David G. Wonnacott. *From Vision to Execution*. URL: http://cs.haverford.edu/people/davew/FVtE.

[16] David G. Wonnacott and Peter-Michael Osera. "A Bridge Anchored on Both Sides: Formal Deduction in Introductory CS, and Code Proofs in Discrete Math". In: *CoRR* abs/1907.04134 (2019). arXiv: 1907.04134. URL: http://arxiv.org/abs/1907.04134.

Designing for Reflective Learning: A Voice-Based Assistant for Intentional LLM Use in Education*

Eric C. Waterhouse[1], Pelumi Abimbola[2], Ayodeji Ibitoye[3] and Babafemi G. Sorinolu[1]
[1]Department of Computer Science
Houghton University
Houghton, NY 14744
{eric.waterhouse26, babafemi.sorinolu}@houghton.edu
[2]Department of Computer Science and Engineering
Mississippi State University
Starkville, MS 39762
pa514@msstate.edu
[3]School of Computing and Mathematical Sciences
University of Greenwich
London, United Kingdom
a.o.ibitoye@greenwich.ac.uk

Abstract

Advancements in AI are reshaping traditional educational practices, presenting both opportunities and challenges. Effective learning demands time, reflection, and active student engagement, beyond the pursuit of immediate results. However, the rapid feedback from large language models (LLMs) risks encouraging surface-level learning and student dependency. This study examines the ethical implications of LLM

use in education and proposes a framework for their intentional and controlled integration. We present a voice assistant powered by an LLM and deployed on a Raspberry Pi, designed to foster reflective, voice-based interactions that preserve opportunities for deep learning while harnessing LLM capabilities. This approach aims to support ethical, dependable, and pedagogically sound human-AI interactions. We anticipate that this tool will encourage student reflection and contribute to more deliberate and meaningful engagement with AI in educational contexts.

1 Introduction

Technological advances have profoundly transformed human life, accelerating communication, increasing access to information, and simplifying daily tasks. These innovations have reshaped sectors such as education and healthcare, yielding significant global benefits. However, one critical concern accompanying technological adoption is the reciprocal adaptation between users and intelligent systems [10]. While these systems are designed to meet human needs, users often alter their behaviors, sometimes in unanticipated or detrimental ways. For instance, the advent of television contributed to increased sedentary lifestyles, while social media platforms have been linked to compulsive engagement and reduced attention spans. Such patterns raise important questions about how students' learning behaviors may evolve in response to large language models (LLMs), warranting careful scrutiny.

Recent studies [9] indicate that LLMs can support learning by delivering personalized feedback, facilitating assessment, and promoting self-directed study. Yet, these same systems pose risks when misused, potentially contributing to cognitive debt, a reduced capacity for independent thinking, problem-solving, and long-term knowledge retention [1, 5]. The widespread release of generative AI tools like ChatGPT has further amplified concerns around academic integrity, with evidence suggesting increased instances of AI-assisted cheating [6]. Moreover, overdependence on LLMs may hinder critical thinking and expose students to misinformation through so-called hallucinations, especially among learners still developing core competencies.

Effective learning requires time, reflection, and sustained effort. The instantaneous outputs of LLMs risk disrupting this process by fostering habits of immediate gratification and passive knowledge consumption. In response, educators have begun employing AI-detection tools (e.g., Turnitin) and redesigning assessments to emphasize higher-order thinking, creativity, and analytical reasoning, skills that are not easily automated. Parallel efforts are emerging to promote AI literacy, equipping students with a deeper understanding of LLM capabilities and limitations, and encouraging ethical, responsible engagement with these tools.

To contribute to this movement, we propose a voice-based solution that encourages more deliberate and structured interaction with LLMs. Unlike conventional chatbot interfaces, our approach positions the LLM as a reflective, tutor-like partner, rather than an on-demand answer engine. We developed a voice-enabled assistant powered by an LLM and deployed on a Raspberry Pi, designed to support meaningful, voice-driven exchanges. By incorporating prompt limits and requiring vocal input, the system discourages overreliance and promotes intentionality, preserving opportunities for deeper learning while leveraging the strengths of AI.

The remainder of this paper is structured as follows: Section 2 reviews related work; Section 3 details our system design and implementation; Section 4 presents early findings and observations; and Section 5 concludes with future research directions.

2 Related Work

2.1 Large Language Models

Large Language Models (LLMs) are increasingly used in education to improve student learning, but studies indicate that their effectiveness varies based on how students interact with them [7]. For example, a student might use AI to write code, but if problems come up, they may not know how to fix them, especially if they've mostly relied on AI instead of learning the basics themselves. But when used well, AI can help students understand new programming ideas, solve errors, and practice skills in a more helpful and engaging way.

In a study done by [4], an LLM-powered robotic agent was used to generate questions based on a professor's lecture content. These questions encouraged critical thinking and class discussion, helping to address limited student-teacher interaction. The study also showed that LLMs create study guides and assess understanding during group or individual learning.

This approach can foster a safer learning environment while helping educators maintain student engagement [2]. Prior studies have also highlighted the use of technologies such as chatbots and robots to enhance student interaction and engagement [2]. Robots, in particular, have the potential to serve as genuine teaching assistants in the classroom, interacting directly with students, or to support remote learners by simulating teacher interactions, thus enabling a truly hybrid learning experience [2].

2.2 Voice Assistants

A 2022 study by [8] found that voice interaction with an AI led to better outcomes than text-based chat, including higher perceived efficiency, lower

cognitive effort, and greater user satisfaction, because speech is a more natural and intuitive mode of communication. Using voice assistants, LLM responses will be delivered in manageable chunks, reducing cognitive load and enhancing learners' ability to recall information.

Research shows that learners benefit more from multimedia instruction when it includes a voice component. To examine whether the type of voice matters, the author [3] compared human and machine generated voices and found no significant difference in learning outcomes between them. Based on this, we argue that limiting students' interaction with the LLM to voice only will encourage deeper focus and attention to the feedback, allowing for better mental processing of the responses before applying the information.

3 System Architecture

To support students' cognitive development and foster effective learning, we configured LLM prompts to align with principles that enhance cognitive processing. The system is designed to produce responses that are clear, decision-supportive, and conducive to improved attention, understanding, and memory retention [11]. The architectural framework of the system is illustrated in Figure. 1.

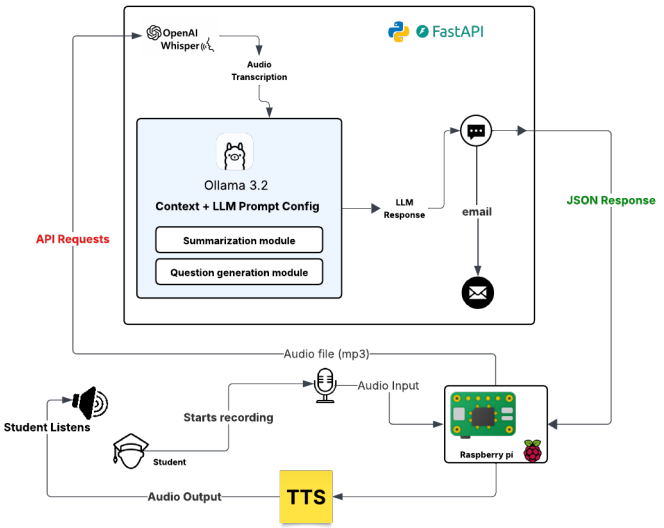


Figure 1: System architectural framework

As depicted in Figure 1, the system integrates multiple functionalities, including lecture recording and transcription, automatic summarization, study question generation, voice-enabled interaction, and email delivery of generated content. Users can either speak directly to the Raspberry Pi or allow the device to passively capture ambient audio. The audio input is recorded and saved locally as an MP3 file, then transmitted to a remote server via an API.

On the server side, OpenAI's Whisper model transcribes the audio to text. Depending on the user's intent, the transcribed input is passed to the LLaMa 3.2 model, which processes the text using predefined prompts and generates a relevant response. This output is returned to the Raspberry Pi in JSON format and simultaneously emailed to the user. Finally, the Raspberry Pi uses the pyttsx3 Python library to vocalize the response through a Bluetooth speaker.

3.1 Design and Implementation

The voice assistant was implemented on a Raspberry Pi 4B, equipped with a 1.5 GHz quad-core ARM processor, 2GB LPDDR4 RAM, dual-band Wi-Fi, Bluetooth 5.0, and four USB ports (two USB 3.0, two USB 2.0), running a 64-bit Debian OS on a 32GB microSD card. The hardware setup included an SSD1306 OLED display (128×64 resolution), a basic push-button for user interaction, a green LED indicator, a USB microphone, and a Bluetooth speaker. A prototype setup, excluding the speaker, is shown in Figure 2.

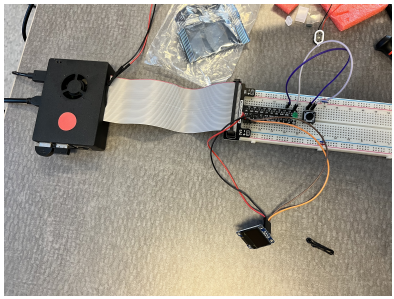


Figure 2: Prototype setup of the voice assistant (excluding the speaker).

Given the computational limitations of the Raspberry Pi, resource-intensive tasks such as transcription and text generation were offloaded to an external server. The server, built using Python and FastAPI, was hosted on an ASUS laptop equipped with an Intel i7-11370H CPU, 16GB RAM, and an NVIDIA RTX 3060 GPU (6GB VRAM). The system employs OpenAI's Whisper-medium model (769 million parameters) for automatic speech recognition (ASR), while

LLaMa 3.2, a publicly available large language model, is used for generating responses. Speech output is handled locally on the Raspberry Pi using the pyttsx3 text-to-speech library.

Prompt engineering was central to system performance and interpretability. Inspired by [4], we tailored prompts for each function. For instance, summarization tasks used the prefix: "Generate a short summary of this lecture:", and study question generation used: "Generate 10 study questions based on this lecture:". For educational guidance, prompts were framed as: "Give a concise response to aid learning and guide the user without revealing the answer immediately." To promote intentional use and reduce overreliance, we introduced a prompt-limiting mechanism. Each user session is capped at five voice queries, after which the system advises the user to reflect before proceeding. Table 1 illustrates the interaction flow.

Table 1: Interaction Flow with Prompt-Limiting Feature.

User Action	**System Response**	**Intended Effect**
Query 1 (Voice)	Provides summary. (4 prompts remaining)	Support reflection and comprehension
Query 2 (Voice)	Generates study questions. (3 prompts remaining)	Promote active learning
...	...	...
Query 5 (Voice)	Provides summary. (0 prompts remaining)	Limit passive overuse
Query 6 (Voice)	"You have reached your query limit for this session."	Encourage reflection and self-study

This interaction flow is designed to balance access to AI assistance with cognitive engagement, encouraging students to reflect on the material rather than rely solely on automated responses. By integrating these constraints, the system aims to support deeper, more intentional learning behaviors

4 Preliminary Results

To assess the functionality and effectiveness of our voice assistant, we conducted a series of preliminary tests focusing on its core features: lecture summarization and study question generation. The system was evaluated using audio recorded from a sample semester presentation describing this research. Our primary objectives were twofold: (1) to demonstrate the system's complete end-to-end operation as outlined in Section 3, and (2) to assess the quality and pedagogical relevance of the generated outputs.

4.1 Designing for Intentional Interaction

A central hypothesis of this study is that interaction design can shape student behavior and promote deeper learning by discouraging dependence on instant AI-generated solutions. To this end, our system incorporates several features explicitly intended to support more deliberate and reflective engagement, challenging the rapid-response paradigm of conventional text-based chatbots. We evaluated the system's design components, namely, the voice-only interface and the built-in prompt-limiting mechanism, through observational and functional analysis. The following elements were particularly salient:

- **Voice as Intentional Friction:** Unlike text interfaces that permit rapid, low-effort querying and copy-paste behaviors, the voice-only interface requires students to mentally formulate their questions and articulate them clearly. This introduces "intentional friction" into the interaction, requiring cognitive effort during query formulation. Observations suggest that this friction slows down the interaction pace, reducing impulsive question submission and promoting thoughtful engagement.

- **Prompt Scarcity via Session Limits:** To discourage overuse and reinforce purposeful AI interaction, the system enforces a cap of five generative prompts per one-hour session. This constraint encourages students to prioritize their queries, reflecting on the most meaningful or challenging aspects of their learning. The intended interaction flow, outlined in Table 1, illustrates how prompt scarcity shapes user behavior toward intentionality.

- **Processing Delay as Reflective Pause:** The system's average response time is approximately 30 seconds, substantially slower than commercial LLM chatbots. While near-real-time, this "forced pause" introduces a reflective buffer between the user's input and the system's response. This delay prevents rapid back-and-forth interactions and instead promotes a rhythm of "question $\rightarrow$ pause $\rightarrow$ reflect $\rightarrow$ answer", giving users space to consider their learning needs before moving forward.

Collectively, these three mechanisms, voice input, prompt scarcity, and deliberate processing delay, form a distinct interaction model that prioritizes intentionality over speed or output volume. Each query incurs a cognitive and temporal "cost," which compels users to think critically about their needs before engaging the system. This aligns with the educational principle that meaningful learning involves time, reflection, and active effort. While full validation of

cognitive outcomes will require a future user study, these early findings confirm that the system's architectural design successfully supports its intended purpose: fostering more thoughtful and disciplined human-AI interactions.

Excerpt from Transcribed Presentation: "...just as if you're having a conversation with a real tutor being able to summarize lectures something that llm technology is good at is you can put in a chunk of text and say summarize this in a different way and it's able to explain things in a different way so that's one of the things that we wanted to do in the classroom. in a way you better understand also you'd receive text-to-speech responses so you wouldn't have to read anything on the screen you just listening to the response that the device is giving you ..."

Generated Summary (from device): The speaker is developing an AI-powered device that uses large language models (LLMs) to assist students with learning. The device takes voice input, transcribes it, and generates text responses based on LLM output. The goal is to create a conversational interface that encourages active listening, deep thinking, and problem-solving skills. The device aims to support traditional LLM use by forcing users to engage in a true conversation, rather than simply copying pre-generated responses. This approach is intended to promote soft skills development and effective learning in the classroom.

Generated Study Questions (from device):

1. How does the proposed device use voice transcription to interact with users?
2. What are some concerns about using LLMs like ChatGPT for academic writing and research?
3. How does the proposed system aim to differentiate itself from traditional LLM use, and what benefits do these differences bring to users?
...
10. How does the device's design promote the development of soft skills, such as active listening and deep thinking?

Figure 3: Example of a generated summary and study questions from a sample presentation describing this research.

4.2 Qualitative Analysis of Generated Content

In addition to interaction design, the quality of the AI-generated output is central to the system's value as an educational support tool. We conducted a qualitative assessment of content produced by the Whisper transcription

and LLaMa 3.2 generation pipeline. Outputs were evaluated for relevance, coherence, and educational utility.

Figure 3 presents a representative example of the system's response to a transcribed segment of the lecture audio. Initial results indicate that the summaries are coherent and contextually appropriate, while the study questions demonstrate a balanced level of difficulty and alignment with core lecture themes. These observations support the feasibility of using the system to reinforce learning and guide student reflection in a non-reductive, supportive manner.

5 Discussion

The preliminary results show that intentional interaction design can meaningfully shape how students engage with AI. By incorporating voice input, prompt limitations, and response delays, our system promotes slower, more reflective use, addressing concerns about cognitive offloading and passive learning often linked to large language models. This approach reframes LLMs as a guided support systems that encourage critical thinking and independent reasoning rather than serving as instant-answer tools. A limitation of our system is that it only works if students choose to engage with it. They can still access traditional LLM interfaces like ChatGPT, which allows them to bypass the intentional friction designed to encourage more thoughtful and reflective learning.

Future work will involve empirical studies to assess the system's impact on learning habits, retention, and student attitudes toward AI, particularly in real-world classroom settings with diverse learners. Overall, this work contributes to a growing conversation about designing AI tools that are not only intelligent, but also ethically and pedagogically aligned.

References

[1] Daniel W. J. Anson. "The impact of large language models on university students' literacy development: a dialogue with Lea and Street's academic literacies framework". In: *Higher Education Research & Development* 43.7 (2024), pp. 1465–1478. DOI: 10.1080/07294360.2024.2332259.

[2] Eleni Dimitriadou and Andreas Lanitis. "A critical evaluation, challenges, and future perspectives of using artificial intelligence and emerging technologies in smart classrooms". In: *Smart Learning Environments* 10.1 (Feb. 2023), p. 12. DOI: 10.1186/s40561-023-00231-3.

[3] Nazmi Dinçer. "The voice effect in multimedia instruction revisited: Does it still exist". en. In: *Journal of Pedagogical Research* 6.3 (May 2022), pp. 17–26. DOI: https://doi.org/10.33902/JPR.202214591.

[4] Shunichiro Ito, Kanae Kochigami, and Takayuki Kanda. "A Robot Dynamically Asking Questions in University Classes". In: *Proceedings of the 2025 ACM/IEEE International Conference on Human-Robot Interaction*. HRI '25. Melbourne, Australia: IEEE Press, 2025, pp. 839–848. DOI: https://dl.acm.org/doi/10.5555/3721488.3721591.

[5] Nataliya Kosmyna et al. *Your Brain on ChatGPT: Accumulation of Cognitive Debt when Using an AI Assistant for Essay Writing Task*. 2025. arXiv: 2506.08872 [cs.AI]. URL: https://arxiv.org/abs/2506.08872.

[6] Victor R. Lee et al. "Cheating in the age of generative AI: A high school survey study of cheating behaviors before and after the release of ChatGPT". In: *Computers and Education: Artificial Intelligence* 7 (2024), p. 100253. DOI: https://doi.org/10.1016/j.caeai.2024.100253.

[7] Matthias Lehmann, Philipp B. Cornelius, and Fabian J. Sting. *AI Meets the Classroom: When Do Large Language Models Harm Learning?* 2025. arXiv: 2409.09047 [cs.CY]. URL: https://arxiv.org/abs/2409.09047.

[8] Christine Rzepka, Benedikt Berger, and Thomas Hess. "Voice Assistant vs. Chatbot – Examining the Fit Between Conversational Agents' Interaction Modalities and Information Search Tasks". In: *Information Systems Frontiers* 24.3 (June 2022), pp. 839–856. ISSN: 1572-9419. DOI: https://doi.org/10.1007/s10796-021-10226-5.

[9] Sahil Sharma et al. "The role of large language models in personalized learning: a systematic review of educational impact". In: *Discover Sustainability* 6.1 (Apr. 2025), p. 243. DOI: 10.1007/s43621-025-01094-z.

[10] Johnny Hartz Søraker and Phillip Brey. "Ambient Intelligence and Problems with Inferring Desires from Behaviour". In: *The International Review of Information Ethics* 8 (Dec. 2007), pp. 7–12. DOI: 10.29173/irie91.

[11] Azmine Toushik Wasi and Mst Rafia Islam. "CogErgLLM: Exploring Large Language Model Systems Design Perspective Using Cognitive Ergonomics". In: *Proceedings of the 1st Workshop on NLP for Science (NLP4Science)*. Association for Computational Linguistics, 2024, pp. 249–258. DOI: http://dx.doi.org/10.18653/v1/2024.nlp4science-1.22.

Introduction to Quantum Computing for Students in Computer Organization Class*

Olivera Grujic
Computer Science
California State University, Stanislaus
Turlock, CA
ogrujic@csustan.edu

Abstract

Quantum computers now exist in prototype form, and their capabilities are expected to grow substantially in the coming years. Even though the quantum computer, once fully functioning, is expected to change the way the world currently operates (even more so than AI shook things up recently), most students are not familiar with the concept or potential application of quantum computing. While some universities are starting to offer courses on quantum computing and post-quantum cryptography, they often require students to take an entire course and complete math and physics prerequisites, which is limiting the number of students who could benefit from learning the quantum computing principles.

This work offered more students an opportunity to explore quantum computing concepts by incorporating a chapter (a week of lectures) on quantum computing in the computer organization class required for undergraduate Computer Science majors, reaching out to over one hundred students. Simple language and analogies, such as rainbows, were used to make quantum theory more accessible to a wider range of students. Student answers on several final exam questions indicated they got the basic understanding of quantum computing concepts and potential applications. The results and experience are summarized in this report.

1 Introduction

The current year, 2025, has been proclaimed the International Year of Quantum Science and Technology (IYQ) by the United Nations (`https://quantum2025.org`). A fully operational quantum computer is expected to transform the way the world operates (e.g. break RSA encryption). D-Wave, Google, Microsoft, and IBM already have working prototypes, and their capabilities are expected to grow substantially in the coming years. Therefore, teaching quantum computing has become increasingly important recently. This need extends beyond academia to industry due to a requirement for new work force development [7] and leads to broader implications to our society [5].

Quantum computing concepts were introduced to over a hundred students in Computer Organization, required to be taken by the CS majors in their junior or senior year at the University in the US. The material was delivered in two lectures during the final week of the semester (Fall 2023 and Fall 2024), each lasting 75 minutes. The first lecture introduced foundational principles such as qubits, superposition, and entanglement. The second focused on gates, circuits, and applications, including demonstrations. The goal was not to comprehensively cover quantum computing, but rather to create a meaningful introduction for students who might never enroll in a quantum-focused course—particularly those in undergraduate CS programs without physics prerequisites.

This experience report is intended as a proof-of-concept for CS faculty exploring entry points for integrating quantum computing into existing undergraduate curricula—particularly for institutions lacking faculty with deep physics or mathematics backgrounds. It is intentionally scoped as a curriculum-based experience report, not a research study; it highlights practical integration rather than proposing new theoretical frameworks or pedagogical models.

1.1 Author Background and Related Work

The interest for learning and teaching quantum computing started after students asked how quantum computer would break RSA encryption. The author participated in workshops aimed at increasing the number of faculty capable of teaching quantum information science and technology (QIST) and participated in the faculty online learning community (FOLC) designed to help educators actively work towards incorporating QIST related concepts in their curriculum. These efforts deepened the author's understanding of quantum computing and supported the development of instructional materials and strategies. More recently, numerous books and platforms such as IBM QISKIT have been developed on quantum physics [8], quantum computing [2],[11], quantum programming [9], quantum cryptography [12], and post-quantum cryptography [3]

with the goal to reach broader audiences.

Some universities are starting to offer courses with the same outreach goal. One study already suggested how to build an entire program in quantum engineering indicating the importance of teaching quantum concepts in the future of college (undergraduate) level education [1]. Another study created a Massive Open Online Course (MOOC) designed for a general audience [10], but the course was completed by approximately 3% of the students (44 out of 1302 took the final exam, mostly those with technical backgrounds and advanced degrees). Another study introduced a post-quantum cryptography course to undergraduate CS majors [4] teaching lattice and elliptical curve based cryptography. In a recent study, quantum programming course was taught to three CS undergraduates and one Chemistry major with a Physics minor [6]. Students reported understanding of quantum problem solving complemented their classical education and found it engaging.

2 Quantum Computing Overview

A brief overview of quantum computing concepts grounded in optics (making them less abstract) is given in this section to illustrate the material taught during lectures and pedagogical approach taken.

2.1 Optics

Light is an electromagnetic wave. Lasers are light sources like flashlights, but create a narrow focused beam of light that is coherent (waves are aligned). Mirrors reflect light. Crystals can act like prisms and bend (refract) light. A shadow is an area where light is blocked. The shadow that a 3D object casts on a surface is essentially a 2D projection of the object's shape.

Seven rainbow colors represent the visible light spectrum, red (the lowest frequency) through violet (the highest frequency). The human eye cannot see lower frequencies (infrared) or higher frequencies (ultraviolet or UV light). The vast majority of the electromagnetic spectrum is invisible to humans, including radio, micro, infrared, and ultraviolet waves, X and gamma rays. Devices such as infrared cameras and X-rays can show parts of the spectrum invisible to humans. Night-vision goggles, like some animals (e.g. cats), see in the dark.

Sunglasses filter light using polarized lenses. Red-blue 3D glasses (anaglyph) allow each eye to receive a different image, which the human brain merges into a single image, simulating a 3D effect. Humans see 24 still images per second, which create the illusion of moving images (movie). Most humans see at a rate of 30 to 60 frames per second. The frame rate of some virtual reality headsets can be too high and cause some people to feel dizzy. Compact Disc

(CD), Digital Versatile Disk (DVD), Blu-ray Disc, and hologram are information storage technologies based on light (optics). Laser printers pass a laser beam repeatedly back and forth over a negatively charged cylinder (drum) to define a differentially charged image.

2.2 Quantum Computer

Quantum bits (qubits) are the basic units of information. Photons are packets of light and can be used as qubits due to their ability to exist in two different states analogous to 0 and 1 of a classical bit. Light is polarized and filters (polarizers) are used to determine the state of the photon. For instance, horizontal filter, which lets (some) light through, adding vertical filter, blocks the light completely, and adding the diagonal filter in between the two lets some (but less) light through. The sequence of the steps appears magical, as the opaque set of filters becomes transparent again after adding another one. EMANIM (`https://emanim.szialab.org/index.html`) simulates wave mechanics.

A qubit can be in both states simultaneously (0 and 1), known as superposition. When the state of qubit is measured, it collapses into one of the two base states, each with some probability. The superposition state gets lost after measuring. The helpful analogy here is the coin spinning mid-air (both heads and tails), but measuring (determining whether the state is heads or tails) requires the coin to stop spinning. Qubits inhabit a higher-dimensional space, visualized using the Bloch sphere, where each point on the sphere represents a possible qubit state. With more qubits, the number of possible states increases exponentially. A qubit can encode exponentially more information than a classical bit, as quantum operations can be used to adjust the superposition's probabilities.

The quantum processor looks like any other computer chip, similar in size and shape. IBM, Google, Microsoft, Intel, and D-Wave approach quantum hardware in their own way. A quantum computer often resembles a chandelier due to the refrigerator used to achieve extremely low temperatures. Qubits need to be cooled to near absolute zero to maintain quantum state. Quantum computation is using quantum circuit, which consists of quantum gates acting on qubits. Quantum gates represent operations that manipulate qubit states. For example, the Hadamard gate puts a qubit in the base state into superposition with an equal chance of each state (and vice versa). The CNOT gate flips the state of the target bit if the control bit is 1. Quantum circuits can be simulated by Quirk (`https://algassert.com/quirk`).

Qubits differ from classical bits because they can be entangled. If we decide to measure two coins spinning in the air, measuring one coin will tell us nothing about the other coin's outcome. However, qubits in superposition can have their measurement outcomes entangled (correlated). Decoherence refers

to the interplay of quantum states (superposition and entanglement) disrupted by interactions with the environment, impacting the reliability of quantum operations. Quantum error correction is a set of techniques used in quantum computing to detect and correct errors caused by interference (noise) and environmental disturbances affecting the qubits.

The Deutsch-Josza and Bernstein-Vazirani algorithms are often considered the simplest and most foundational. They demonstrate the quantum speedup without involving complex mathematical concepts and provide a solid foundation for learning more advanced quantum algorithms, such as Grover's and Shor's. Quantum computing promises to revolutionize industries by tackling complex problems currently intractable for classical computers. It is also being explored in AI/machine learning. Some key applications include: drug discovery, cybersecurity, financial modeling, material science, manufacturing, energy, and logistics.

3 Outcomes of Delivery

Teaching methods used to convey the information listed in section 2 are discussed, and (anonymized) student learning impressions are listed in this section. As student reflections and exam responses were collected anonymously as part of normal classroom evaluation, no Institutional Review Board (IRB) approval was required.

3.1 Teaching Methods

The course was taught in two sections in Fall 2023 and three sections in Fall 2024. The total number of students enrolled across the five sections was 109. This is a significant number of students to reach, given the small class sizes and student-to-faculty ratio that allows for interaction.

Teaching about a viable alternative technology provided additional perspective to the architecture currently used, strengthening and complementing the core concepts of the course (e.g. bits vs qubits, classical vs. quantum circuits).

Many abstract concepts were taught by analogy, as discussed in section 2 (e.g. light, shadow, coin) with an intention to make them more understandable. Simplified analogies like the coin model do not fully capture quantum formality, but were used intentionally to make ideas approachable in a non-major course. Hands on exercises using light polarizers and simulators (e.g. EMANIM and Quirk) were used to aid in active learning.

3.2 Student Participation

Collaborative learning was encouraged throughout. The author therefore asked students to take ownership and share their own examples of the concepts discussed in class. Some students participated. For instance, the example of 3D glasses came from a student. The figure that humans only see approximately 0.0035% of the entire electromagnetic spectrum was looked up by another student.

A student recalled watching a "60 Minutes" episode about quantum computing and shared the link with the class. The episode provided a broader overview of quantum computing, its potential and some challenges associated with it. The camera crew was able to showcase the quantum computers developed by IBM and Google by gaining access to their facilities, which are not available to public.

They also interviewed some of their scientists. Physicist and best-selling author Michio Kaku was one of them. He used the analogy of a maze to explain how quantum computers could outperform classical computers by exploring all possible paths simultaneously, leading to a much faster solution. This motivated additional class discussion.

3.3 Student Learning

The final exam contained two to three questions related to quantum computing worth a small percentage (12%) of the exam grade. Since there were five class sections, there were five questions in total that were shuffled across the five versions of the exam so no section got all the same questions. Four questions were fill in the blank and one was a free response.

The questions were designed to gain insight into whether students were engaged and if they retained the basics. Most students answered the questions correctly. The results are illustrated in Figure 1 (number of total responses) and Figure 2 (percentage of correct answers).

Question 1: Quantum computers use qubits that can be in multiple states simultaneously. It was answered by 85 students across four sections. 84 out of 85 students answered it correctly (98.8% accuracy). The idea was to verify the students heard the lecture. Apparently there was a student who missed the lecture.

Question 2: A qubit can be in multiple states at the same time. This is called superposition. It was answered by all 65 students correctly (100% accuracy).

Question 3: The state of quantum bit is determined by the spin of an electron. It was answered by 19 out of 20 students correctly (95% accuracy).

Question 4: Decoherence causes uncorrectable errors; advanced error-correction

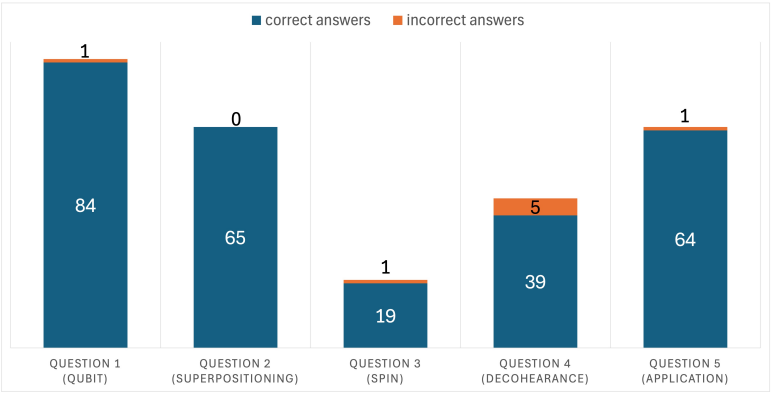


Figure 1: Number of student responses to exam questions.

algorithms have been applied to this problem. It was answered correctly by 39 out of 44 students (89% accuracy). This confirms the expectation that some students would miss the more advanced concepts.

Question 5: What are some of the areas in which quantum computers can be applied? This question was posed to 65 students across three sections, and was answered correctly by 64 of them (98.5% accuracy). One student left the response blank. The answers varied significantly among students.

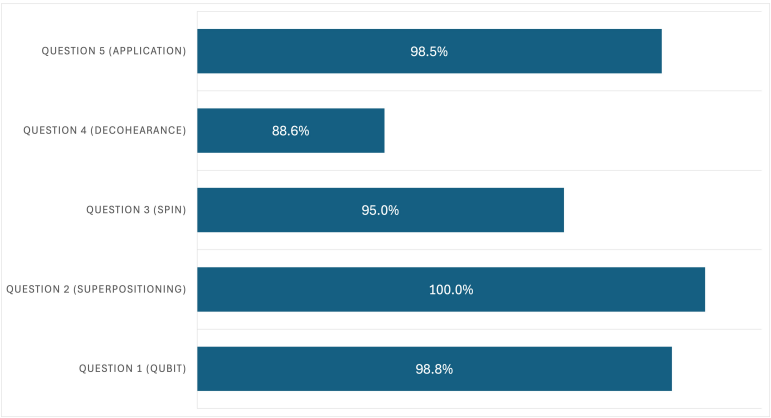


Figure 2: Percentage of correct answers to exam questions.

The students understood quantum computers can be applied in AI, cybersecurity and drug discovery, as discussed in class. Notably, two thirds of

students (43 out 65) answered that quantum computers have potential application in cybersecurity, indicating the potential to break RSA encryption, crack passwords and generate true random numbers. Half of the students (33 out of 65) indicated the application in AI and data search. Almost one third of students (19 out of 65) listed application in drug discovery, protein folding, or medicine in general. The answers are in line with the material taught in class and with current events and experiences in our society.

Some students listed other commonly mentioned applications: financial modeling, material science, logistics and supply chain optimization, optimization problems, weather forecasting, and energy research, as indicated in Figure 3.

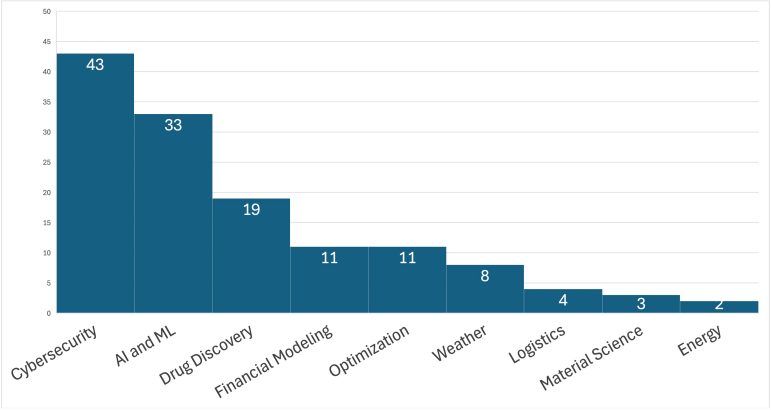


Figure 3: Number of responses per potential application of quantum computers.

3.4 Student Answers

Some students' answers to question 5 (potential applications of quantum computing) were more creative, listing potential applications in aerospace, agriculture, architecture, gambling, gaming, optical computing, personalized medicine, quantum simulations (of subatomic particles), telecommunications and networking, traffic planning and even understanding of the Universe. Here, the author lists a few anonymized student answers for illustration purposes.

Agriculture: "One area where quantum computers can be applied would be the agriculture sector through a variety of methods such as improving the managing and health of crops by analyzing the data with quantum computing or having weather predictions calculated by quantum computing that are more accurate which would allow crops to be more efficiently planted and chosen".

Architecture: “Another area where quantum computers can be applied would be towards the architecture sector with architecture designs generated by quantum computing that are significantly more complex and detailed as opposed to designs made by the current computing. The quantum computing can also make designs more efficient not just from a building perspective but also a utility with more efficient energy usage or stronger stability of the building”.

Gambling: “Quantum computers can be applied in casinos to prevent a player from winning too much too quickly, thereby creating a ’limit’ to how much they can win at a time”.

Personalized Medicine: “It can also be used in this area by analyzing the DNA of individual patients, and from that, derive the best drug concoctions for each person, bringing to the table a whole new level of personalization that goes beyond simply diagnosing patients based on symptoms or x-rays alone”.

Space Exploration: “I was reading something saying that AI figured out what Mars atmosphere was like, something it was estimated humans to take years took AI weeks. Now imagine AI had the power of quantum computers”.

Traffic planning: “Quantum computers can be used in procedures that require multiple calculations, from protein folding to traffic analysis and planning, just like the very first tabulating machine, we’re making computers that can do math faster”.

Universe: “Because they are quantum computers, they can give us insight about how the universe works (since the universe can be viewed as a quantum computer)”.

The assessment used recall-level questions embedded in the final exam to gauge basic retention. While not a measure of conceptual mastery, this provided initial feedback on student engagement and comprehension. Future work may explore pre/post testing and concept inventories.

4 Conclusion

This experience — from limited background to lecture delivery — proved worthwhile. The integration reached over one hundred CS students, who likely wouldn’t have taken a quantum computing course.

The small class sizes (20–24 students) enabled student interaction. Students and professor found the quantum computing topic interesting and engaging.

The students provided correct and creative answers to relevant exam questions. They understood quantum computers can be applied in AI, cybersecurity and drug discovery among other areas. They also proposed some novel applications.

The initial feedback on student engagement and comprehension inspired the author to create and teach the quantum computing course.

Acknowledgment

The author thanks everyone involved with QIST and FOLC for their support.

References

[1] A. Asfaw et al. "Building a Quantum Engineering Undergraduate Program". In: *IEEE Transactions on Education* 65.2 (2022), pp. 98–105. DOI: 10.1109/TE.2022.3144943.

[2] C. Bernhardt. *Quantum Computing for Everyone*. MIT Press, 2020.

[3] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer, 2010.

[4] T. J. Borrelli, K. D. Nance, and L. V. Snyder. "Post-Quantum Cryptography Course". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. 2024. DOI: 10.1145/3626252.3630823.

[5] M. F. J. Fox et al. "Preparing for the quantum revolution: What is the role of higher education?" In: *Physical Review Physics Education Research* 16.2 (2020), p. 020131. DOI: 10.1103/PhysRevPhysEducRes.16.020131.

[6] J. T. Guerin. "Creating a Quantum Programming Course from Scratch". In: *Proceedings of the SIGCSETS 2025 Conference*. 2025, pp. 409–415.

[7] C. Hughes et al. "Assessing the Needs of the Quantum Industry". In: *IEEE Transactions on Education* 65.4 (2022), pp. 377–383. DOI: 10.1109/TE.2022.3153841.

[8] B. Ishak. "Quantum physics: What everybody needs to know, by M. G. Raymer". In: *Contemporary Physics* 59.1 (2018), pp. 99–100. DOI: 10.1080/00107514.2017.1403467.

[9] E. R. Johnston, N. Harrigan, and M. G. Haner. *Programming Quantum Computers*. O'Reilly Media, 2019.

[10] J. Liu and D. Franklin. "Introduction to Quantum Computing for Everyone". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023)*. 2023. DOI: 10.1145/3545945.3569836.

[11] E. Rieffel and W. H. Polak. *Quantum Computing: A Gentle Introduction*. MIT Press, 2011.

[12] T. Vidick and S. Wehner. *Introduction to Quantum Cryptography*. Cambridge University Press, 2024.

Sentiment Analysis on U.S. Politics with BERT Models*

Yi Liu, David Mears, Evan Dunnam
Department of Information Systems and Computer Science
Georgia College and State University
Milledgeville, GA 31061
yi.liu@gcsu.edu, david.mears@bobcats.gcsu.edu, evan.dunnam@bobcats.gcsu.edu,

Abstract

This paper explored different Bidirectional Encoder Representations from Transformers (BERT) models on sentiment analysis to American political discourse using Reddit as a primary data source. This study leveraged Reddit's accessible API and active political communities to extract a large volume of opinion-rich text. Then entity-level sentiment analysis was applied to distinguish sentiment directed at specific political figures, issues, or policies. Three BERT models-BERT-Base, DistilBERT, and XLM-T—were evaluated for their accuracy based on title-comments data and comments-only data. The results confirmed that BERT models performed better with appropriate contexts, even just with a single sentence. The results also showed that XLM-T performed the best among the three BERT-based models, and was selected for further fine-tuning. A manually labeled dataset of Reddit comments was used to fine-tune and evaluate the model. Despite challenges such as limited positive sentiment samples and hardware constraints, the fine-tuned XLM-T model demonstrated measurable performance improvement using just over 3,000 comments. This indicated that the model could serve as a powerful tool for political sentiment analysis, particularly in domain-specific vocabulary, and evolving political contexts. This research also showed a practical approach to developing natural language processing

(NLP) applications under limited resource conditions and laid the foundation for a future web-based tool to visualize political sentiment trends over time and across topics.

1 Introduction

The goal of this research project is to apply BERT-based sentiment analysis(SA) methods to American political discourse by analyzing opinions from social media platforms, with a particular focus on Reddit. Prior studies have frequently used Twitter (now X) or news as a data source for political SA [5, 3, 1] with different methodologies such as TF-IDF[11], SVM[13], Bayesian Network [14], or deep learning models, including CNN [10], RNN [3, 2], and LSTM [16, 1]. However, BERT models have not been applied and compared based on political SA with Reddit's unique context-rich conversations.

BERT is a pre-trained language model developed by Google AI and introduced by Devlin et al. [7]. It is identified as a major breakthrough in natural language processing (NLP) by enabling bidirectional context understanding with self-attention mechanism. BERT is trained using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), allowing it to generalize language representations and be able to capture deeper contextual relationships, and therefore, only requiring a few epochs of training and relatively small amounts of labeled data to achieve good results for downstream NLP tasks.

Compared with Twitter, Reddit is organized using subreddits, which form Forum-style discussions (titles + comments), and allows context-based longer discussion. Reddit is becoming an important social platform with very accessible API and a high volume of politically engaged discussions. BERT models' capability that captures long-range dependencies and bidirectional context makes them a good choices for Reddit data.

Moreover, BERT can be easily fine-tuned for various NLP tasks by simply adding just one additional output layer with a small amount of task-specific training data. This pre-train with fine-tune paradigm makes BERT highly efficient, especially in settings with limited resources. As a result, there are many fine-tuned BERT-based models on different domains and datasets.

To capture political sentiment in this project, three BERT-based sentiment analysis (SA) models BERT-Base [7], DistilBERT [15], and XLM-T [4] were selected and evaluated using Reddit's forum-style data. After testing and comparing these models, the study identified the best-performing one and proceeded to fine-tune it under limited computational resources, using data that included up-to-date political figures and issues. This efficient fine-tuning process enabled the model to adapt the quickly evolving political contexts

characterized by domain-specific vocabulary, nuance, and sarcasm.

The ultimate goal of the study is to develop a fine-tuned BERT model that can serve as the foundation for a daily political SA and adaptive visualization tool. This research also shows a practical approach to implementing AI applications at the undergraduate level by leveraging open-source large language models (LLMs) and access to only one or two GPUs.

The study acknowledged several challenges to political SA. These included difficulties in capturing nuance and sarcasm, particularly when up-to-date political knowledge was required, as well as the low accuracy in detecting neutral sentiment. Addressing these challenges will remain for improving model performance in future iterations.

2 Methodology

2.1 Political Sentiment Analysis

Sentiment analysis refers to the study of people's emotions, opinions, attitudes, or their feelings on a subject. Researching the sentiments of a group has many useful applications, from a political campaign measuring how voters feel about key issues to stock analysts measuring the predicted success of a company. The process of such analysis can be highly complex, especially if those conducting the study choose to look for sentiments on a wide range of entities. Areas of study like natural language processing have laid the groundwork for analyzing public opinion, and the proliferation of social media worldwide has given researchers the ability to look at much larger groups than ever before, allowing for a more complete picture of a subject's approval.

SA can be done at one of three levels [11]-document, sentence, and aspect/entity level. The document level of analysis refers to the classification of a full document as either positive or negative. At this level, it is assumed that the document being analyzed contains sentiment on only a single entity. Being the simplest of the three, document level analysis is the easiest to implement, but can leave out large amounts of information that might otherwise be useful in the analysis. Simplifying documents to such a level is a dangerous way to try and predict or observe opinions on an entity. A step up from this is sentence-level analysis, which breaks the document down by each sentence. From there, each sentence can be recorded on its sentiment of the entity in question, that being positive, negative, or neutral. It is important to include the neutral sentiment at this level, as not every sentence will offer an opinion and may merely state a fact or be unrelated entirely. Analysis can become complicated at this level. A metaphor used may go unnoticed and recorded as a neutral sentiment, even though a reader would recognize it to disparage the entity in question. Missing contextual meaning and mixed sentiment are also

two top problems, For example, a phrase like “Putin loves it" literally sound positive, but it can carry negative sentiment if the context is related to a U.S. policy or action which is framed as beneficial to an adversarial figure (like Putin). A phrase of a mixed-sentiment sentence could be “The president’s tax cuts boosted the economy in the short term, but they increased the national debt significantly". The final classification level is entity/aspect. For consistency, this will simply be referred to as entity level SA. This level breaks down a sentence and looks closer at what entity is being described. For example, the sentence “Although his tax cuts helped some businesses, but his leadership could deeply divide the country.” exemplifies this level of classification. In this sentence, there are two “sentiment-target pairs" expressed. The first is that the policy “tax cut" is positive by some businesses, but the second expressed negative sentiment about “leadership". The entity level classification is made around this sentiment-target pair breakdown, allowing the analysis to break down the larger entity of a particular policy or figure into multiple smaller entities like tax cut policies or leadership, which is much more useful if it were being used to help improve insight in political sentiment tracking or election analytics, and is more actionable than general approval/disapproval ratings.

Our research focuses on developing a structured approach to aspect and entity-level sentiment analysis (SA) tailored specifically to political texts from social media platforms like Reddit. Compared to general-purpose SA, political SA presents unique challenges, including sarcasm, subtlety, domain-specific vocabulary, and evolving political contexts. Therefore, the ability to fine-tune a model when needed, especially under resource constraints, is important for us to detect which aspects of a political subject are being praised or criticized.

2.2 Data Collection

SA based on social media is commonly approached in two phases: relevant data is first collected, followed by the extraction of sentiment from that data. One of the most promising social media outlets for data collection is Reddit, which offers large sets of data for free in a format that can be preprocessed and analyzed.

Data collection usually involves either manual or automated compilation of text. Fully automated scraping techniques are powerful, as they can collect large volumes of text from online sources. In our research, we automatically scraped post titles and their associated comments from the subreddit r/politics using Reddit’s API as needed. This subreddit is one of the largest political communities on Reddit. A post title scrapped can be “Hundreds of Canadian steel workers hit with layoffs as Trump’s tariffs squeeze industry", and an associated comment can be “Better yet they could move to America.".

Once the data were collected, preprocessing steps, such as tokenization,

normalization, and filtering, were applied to clean and prepare the text for analysis. Emojis, images, and other non-text elements that the model may not interpret were removed. However, if such unreadable content (e.g., GIFs or emojis) is important for understanding sentiment, replacing them with descriptive text that the model can interpret during preprocessing may provide a better solution [8].

2.3 BERT Models for Sentiment Analysis

Transformer models were introduced by Vaswani et al. from Google AI [16]. They are entirely based on a mechanism called self-attention. These models addressed the limitations of previous sequence models, particularly RNNs and LSTMs, which rely on recurrence to process sequences. Both RNNs and LSTMs are slow and struggle with long-range dependencies as well as inefficient parallelization. With the self-attention mechanism and an encoder–decoder structure, Transformer models capture long-range dependencies between words regardless of their distance in the text [16]. They can also process words in a sentence in parallel, making them faster to train than RNNs and LSTMs, and therefore scalable to large datasets. Transformer models have outperformed classical models in many natural language tasks [16, 17, 5], marking a paradigm shift in NLP fields since 2018. As a result, Transformer models are now the backbone of systems like GPT and BERT.

BERT was introduced by Devlin et al. from Google AI [7] in 2018. It is a Transformer-based model that deeply considers the context of a word in all layers by incorporating both the words before and after it in a sentence. BERT was trained on two massive datasets: BookCorpus (800M words) and Wikipedia (2,500M words). By using two unsupervised tasks—MLM and NSP—BERT was pre-trained bidirectionally on full input sequences using the Transformer's encoder. This enabled it to better capture the meaning of ambiguous words and understand complex sentence structures. After training, the BERT-Base model consisted of 12 layers, 12 attention heads, and 110 million parameters, while the BERT-Large model consisted of 24 layers, 16 attention heads, and 340 million parameters. Both models set new state-of-the-art benchmarks across eleven natural language processing tasks, including GLUE, MultiNLI, SQuAD, and others [7].

Although training BERT is computationally expensive and requires significant resources for inference, its bidirectional training allows the model to capture richer contextual information. Moreover, the MLM and NSP objectives equip BERT with general linguistic knowledge, making it adaptable to a variety of downstream tasks using much smaller labeled datasets and significantly fewer computational resources.

Compared to BERT, traditional NLP models suffer from two major limi-

tations: the need for extensive training for each task and the requirement to carefully design task-specific architectures. Both issues require expensive computational resources and experienced personnel. In contrast, BERT only needs to be pre-trained once and can then be fine-tuned for a variety of tasks, significantly reducing the resources needed to build high-performance NLP models.

Devlin et al. [7] particularly highlighted BERT's key strength: its ability to be fine-tuned for a wide range of NLP tasks with minimal architectural modifications. By using bidirectional context through the self-attention mechanism, BERT maintains a consistent architecture across various NLP tasks. There is minimal difference between its pre-trained structure and the final task-specific model. This capability enables BERT to achieve state-of-the-art performance across eleven benchmark tasks, including text classification, question answering (QA), and natural language inference, with much less effort. As stated in the paper, "The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of task" [7]. The authors further demonstrated that, compared to pre-training, fine-tuning was computationally inexpensive. All results for the eleven NLP tasks were achieved within a few hours on a GPU, or in at most one hour on a single Cloud TPU [7].

3 Experiments

3.1 Model Selection

To perform political SA, training a model from scratch for our research was not feasible due to the substantial computational resources and large-scale data requirements involved. Therefore, pre-trained BERT models designed for SA were selected for their strong contextual understanding and ease of fine-tuning.

There are many variants of the BERT model available on Hugging Face, including both general-purpose models and domain-specific versions tailored for fields such as finance, law, and social media. After evaluation, three BERT-based models were selected for further analysis based on their suitability for SA tasks, computational efficiency during fine-tuning, the datasets used in their pre-training, and overall availability: BERT-Base [7], DistilBERT [15], and XLM-T [4]. All three models are publicly available on Hugging Face.

3.1.1 BERT-Base Model

As mentioned above, BERT-Base was trained on approximately 3.3 billion words from English Wikipedia and Google's BookCorpus. It consists of 12 layers of transformer encoder, 768 hidden units, and 12 heads of self-attention, totaling 110 million parameters[7]. BERT-Base has become a widely used

base model because of its strong generalization ability and robust performance across a range of NLP tasks.

3.1.2 DistilBERT Model

DistilBERT was developed by Hugging Face using a technique called knowledge distillation, where a smaller model (the "student") was trained to replicate the behavior of a larger, pre-trained BERT-Base model (the "teacher"). It minimizes a combined objective that includes the soft target loss from the teacher, the MLM loss, and the cosine embedding loss between hidden states.

The primary goal of DistilBERT is to reduce the size, inference time, and resource requirements of large language models, enabling wider adoption while retaining most of BERT's language understanding capabilities and performance. DistilBERT contains only 6 Transformer layers (compared to 12 in BERT-Base), 2 attention heads, and a hidden size of 768. It removes NSP objective, which has been shown to contribute little to downstream performance. The model was trained using a multi-objective loss that includes MLM loss, distillation loss, and cosine embedding loss. As a result, DistilBERT is 40% smaller, 60% faster, and retains 97% of BERT's language understanding capabilities [15], making it well-suited for real-time applications and resource-constrained environments, while still delivering strong performance on NLP tasks.

3.1.3 XLM-T Model

XLM-T is an extension of the RoBERTa model [12], which was developed by Facebook AI and introduced by Liu et al. RoBERTa retains the original Transformer encoder architecture of BERT but makes several key modifications to the pretraining process. Notably, it removes the NSP objective, which was found to be unnecessary and even slightly detrimental to downstream performance. Additionally, RoBERTa is trained on 10 times more data than BERT, using a much larger and more diverse corpus including CommonCrawl and OpenWebText. The model is also trained for longer with larger mini-batches and learning rates, which helps it better capture long-range dependencies and generalize effectively. Moreover, RoBERTa applies dynamic masking, where the masked tokens are selected during its training time rather than being fixed before training, as in BERT-Base. The results show that RoBERTa outperformed BERT on several benchmarks, including GLUE, RACE, and SQuAD, without modifying the underlying model architecture.

XLM-T was pre-trained on 198 million tweets based on the architecture of RoBERTa from Facebook AI [6]. By adapting to the informal linguistic characteristics of Twitter data, XLM-T achieves strong performance on several benchmarks [4]. XLM-T has 270 million parameters.

3.1.4 Data for Model Testing

For the purpose of evaluating the selected models, we collected 3,143 comments, nearly all of which came from posts in the r/politics subreddit. After data collection, the data set was pre-processed to remove noise and standardize inputs. However, for Transformer-based models such as BERT, aggressive pre-processing, such as stemming or stopword removal, can disrupt the raw text structure and semantic context, on which BERT models rely [17]. Therefore, we limited pre-processing to the removal of comments from auto-moderation bots, GIFs/images, emojis, and irrelevant characters. Then each sentence was tokenized before being passed to the models.

To evaluate the performance of the three BERT-based models, we manually labeled each comment as positive, negative, or neutral. The labeled data were then fed into the models, and a confusion matrix was generated for each. A confusion matrix is a widely used evaluation tool in classification tasks that summarizes a model's predictive performance by comparing its outputs to the true class labels. Each row of the matrix represents the actual class, while each column corresponds to the predicted class, providing a detailed view of both correct and incorrect classifications.

While testing the performance of the models, two input configurations were evaluated: comments only and combined data (original title + comment). Each model was tested twice—one using only the comment as input, and the other using the combined title and comment. The combined input consistently yielded better performance, especially on negative comments, suggesting that contextualizing comments with their associated title enhances the model's understanding.

3.1.5 Model Fine-Tuning

After evaluating the three BERT-based models, the XLM-T model showed the best overall performance. As a result, it was selected for continued fine-tuning. A total of 3,143 comments were manually labeled for this phase, and the dataset was split into 90% for training and 10% for testing. The training arguments followed Hugging Face's recommendation for fine-tuning BERT-based model. The model used a batch size of 2 per device, with gradient accumulation steps set to 2, simulating a batch size of 4 while maintaining compatibility with limited GPU memory. A learning rate of 2e-5 was chosen, which is a commonly used value for BERT fine-tuning, ensuring stable convergence. While this is a relatively small dataset for training, the results did show improvement based on the confusion matrix generated from the 10% test set.

3.2 Results

The following confusion matrices showed the performance of the three BERT models for both combined data (title and comment) and comment only. Under each confusion matrix was the accuracy score, or the percent of data that the model processed correctly. Table 1 compared the performance of the XLM-T model on two input configurations: combined data (title + comment) and comments only. The model accurately identified 2,367 out of 2,395 negative inputs and 434 out of 545 positive inputs, with the higher accuracy of 89%. In contrast, with only comments, the model accurately identified 2,100 negative and 422 positive inputs with accuracy of 82%, demonstrating the value of contextual enrichment for improving sentiment detection in social media data with BERT models.

However, despite being the most accurate model overall (89% accuracy on combined data), it struggled to detect neutral comments. Only 5 out of 203 actual neutral comments were correctly classified in the combined data setting, and 48 were correctly predicted as neutral in the comments-only setting. The correct classification rate for neutral comments was much lower than that for positive or negative comments across all three models.

	Combined Data			**Comments Only**		
	Predicted Negative	Predicted Neutral	Predicted Positive	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	2367	3	25	2100	151	144
Actual Neutral	192	5	6	126	48	29
Actual Positive	109	2	434	81	42	422
	Accuracy Score: 89%			**Accuracy Score: 82%**		

Table 1: Confusion matrices and accuracy scores for XLM-T model on combined data and comments only

	Combined Data			**Comments Only**		
	Predicted Negative	Predicted Neutral	Predicted Positive	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	2177	218	0	2041	354	0
Actual Neutral	166	37	0	130	73	0
Actual Positive	269	276	0	167	378	0
	Accuracy Score: 70%			**Accuracy Score: 67%**		

Table 2: Confusion matrices and accuracy scores for DistilBERT model on combined data and comments only

Tables 2 and 3 show the sentiment classification performance of DistilBERT and BERT-Base using combined data and comments only. Unfortunately, DistilBERT failed to predict any positive sentiment. Overall, XLM-T performed best. BERT-Base followed with accuracy scores of 79% and 67%, while DistilBERT had the lowest performance, scoring 70% and 67%. All of the BERT-based models achieved higher accuracy scores when tested with the combined input compared to using comments alone.

	Combined Data			Comments Only		
	Predicted Negative	Predicted Neutral	Predicted Positive	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	2337	17	41	1705	292	398
Actual Neutral	196	4	3	106	41	56
Actual Positive	359	34	152	118	67	360
	Accuracy Score: 79%			**Accuracy Score: 67%**		

Table 3: Confusion matrices and accuracy scores for BERT-Base model on combined data and comments only

As mentioned, the XLM-T model achieved the highest overall performance and was therefore chosen for further fine-tuning. Combined data were used for both training and testing, as this configuration had previously demonstrated the highest accuracy. Using the 10% of data reserved for testing, a confusion matrix was generated, as shown below.

Post Training, XLM-T Model, Combined Data			
	Predicted: Negative	Predicted: Neutral	Predicted: Positive
Actual Negative	232	2	4
Actual Neutral	12	6	2
Actual Positive	6	0	51
Accuracy Score: 92%			

Table 4: Confusion matrix for post-training XLM-T model on combined data

3.3 Challenges Encountered

As previously stated, data can be collected using APIs from the subreddit r/politics, and then manually labeled for model evaluation and training. However, the majority of the comments on this subreddit were negative. While this is not a major issue for an application that tracks sentiment values over time, it could be problematic for training. If there is a very small amount of data associated with a particular sentiment label, the model may become biased toward the more frequent labels. To prevent this, the sentiment distribution in the training data was controlled. More positive data were added to the original dataset than naturally occurred. After the proportion of positive comments was raised to around 17%, the model was then fine-tuned.

The misclassification of neutral sentiments was another issue across all the models tested, highlighting one of the core challenges in political SA. All three models performed poorly on neutral sentiment detection, particularly when using the combined inputs. After fine-tuning, the model identified only 6 out of 20 neutral instances in the test set, indicating that it still struggled to distinguish neutral expressions from weakly negative ones. Several factors may contribute to this issue. First, the model may fail to capture nuance or subtle cues that define neutrality, especially in short or informal texts like user comments. Second, in the combined input setting (title + comment), the comment

may be neutral, but if the associated title implies disapproval, the model could be influenced by the overall negative tone, leading to misclassification.

To enhance future performance, particularly in identifying neutral sentiment, this research will continue to focus on augmenting the training data with a greater number of neutral comments, with the goal of reducing the misclassification of neutral expressions.

Training the XLM-T model revealed hardware limitations, specifically, an RTX 3070 GPU with only 8GB of VRAM, which led to a system crash in mid-training. As a result, training parameters had to be adjusted to reduce computational demands. These adjustments included reducing the batch size from 8 to 2, enabling half-precision floating-point (fp16) computation, and applying gradient accumulation over 2 steps. While these modifications allowed the model to be trained successfully under resource constraints, they may have compromised the overall quality of training. So increasing computational power, such as acquiring a more powerful GPU or using multiple GPUs, is another direction for achieving a higher-quality model. With additional resources, it becomes possible to increase the batch size, extend the input sequence length, and even apply dynamic learning rate [9] to improve model performance.

4 Conclusion

This research explored the application of BERT-based models for political sentiment analysis using data collected from Reddit, a platform known for its forum-style discussions. While prior studies have analyzed political sentiment using data from Twitter or news sources with models such as SVM, Bayesian networks, RNNs, and LSTMs, this work uniquely applied and compared BERT, DistilBERT, and XLM-T using Reddit data. BERT's bidirectional context understanding and ease of fine-tuning make it a strong candidate for political sentiment analysis, which involves continuously evolving domain-specific knowledge. The study evaluated the performance of these models using a manually labeled dataset from Reddit. Despite challenges such as limited computing resources and the complexity of sarcastic or subtle political expressions, the results showed that the XLM-T model was efficient and easy to fine-tune, making it adaptable for daily political sentiment tracking. This work also demonstrated a practical approach to implementing advanced NLP techniques under resource constraints, laying the groundwork for future tools to visualize and analyze sentiment trends in political discourse.

References

[1] M. Z. Ansari et al. "Analysis of political sentiment orientations on Twitter". In: *Procedia Computer Science* 167 (2020), pp. 1821–1828. DOI: 10.1016/j.procs.2020.03.201.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *CoRR* abs/1409.0473 (2014). URL: https://arxiv.org/abs/1409.0473.

[3] K. Baktha and B. K. Tripathy. "Investigation of Recurrent Neural Networks in the Field of Sentiment Analysis". In: *2017 International Conference on Communication and Signal Processing (ICCSP)*. Chennai, India: IEEE, 2017, pp. 2047–2050. DOI: 10.1109/ICCSP.2017.8286763.

[4] Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. "XLM-T: Multilingual Language Models in Twitter for Sentiment Analysis and Beyond". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2022, pp. 258–266.

[5] Francesco Barbieri et al. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020), pp. 1644–1650.

[6] Alexis Conneau et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020. URL: https://arxiv.org/abs/1911.02116.

[7] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* 1.1 (2019), pp. 1–12.

[8] Bjarke Felbo et al. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. 2017, pp. 1615–1625.

[9] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 328–339. URL: https://aclanthology.org/P18-1031.

[10] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Vol. 1. 2014, pp. 1746–1751.

[11] Bing Liu. *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers, 2012.

[12] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *arXiv preprint arXiv:1907.11692* 1 (2019), pp. 1–13.

[13] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. “Thumbs up? Sentiment classification using machine learning techniques”. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. 2002, pp. 79–86.

[14] Pablo A. Ruz Gonzalo A. and Aldo Mascareño. “Bayesian Constitutionalization: Twitter Sentiment Analysis of the Chilean Constitutional Process through Bayesian Network Classifiers”. In: *Mathematics* 10.2 (2022), p. 166. DOI: 10.3390/math10020166. URL: https://doi.org/10.3390/math10020166.

[15] Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108* 1.1 (2019), pp. 1–8.

[16] Ashish Vaswani et al. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 5998–6008.

[17] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 38–45.

Prevalence of Autism Spectrum Characteristics in Students Taking Undergraduate Computing Courses *

Rachel Michaela Mettenbrink[1] and Stephen Cooper[2]
School of Computing
University of Nebraska-Lincoln
Lincoln, NE
[1] rami@unl.edu
[2] stephen.cooper@unl.edu

Abstract

This study tests for the prevalence of autism spectrum condition (ASC) traits measured by delivering the Autism Spectrum Quotient (AQ) to a population of undergraduate computer science students. We examine the relationships between AQ scores and students taking undergraduate computer science classes, sex, socioeconomic status, and parents in the computing industry. Additionally, we compare the AQ scores of our participants with publicly available control data. As in previous studies of ASC, we saw a noteworthy increase in AQ score amongst participants over the average general population rate reported in the literature. We saw a statistically significant increase in mean AQ score across most demographic metrics, including both higher male and female means and higher means in the topmost socioeconomic classes. This supports existing evidence for an increase in the prevalence of ASC amongst the general population, as well as for a relationship between ASC and participation in computing.

1 Introduction

As computer science (CS) continues to grow as a desirable field of undergraduate study, considerable attention is being given to understanding and improving the educational experiences of all who are represented within the student body. Whilst the existing body of research already represents a wide range of individuals, there is little consideration in the computer science education literature given to neurodivergence and the impacts it has on computing education.

Neurodevelopmental disorders such as the autism spectrum disorders (ASD) have received significant attention in medical literature in recent years as awareness of the disorders has increased. Diagnostic tools have improved, and stigmas associated with diagnoses and characteristics of said disorders have steadily reduced [12].

Computer science literature has yet to address these disorders at large. Although research into neurodivergent computing has started to appear (e.g., [9] [7]), there is currently no literature demonstrating a relationship specifically between the pursuit of undergraduate computing education and neurodivergence, despite a general conception amongst some computer scientists that such a relationship exists. Without scientific literature documenting such a relationship, this "common knowledge" is nothing more than folk wisdom, which we know to frequently be incorrect. The lack of evidence presented thus far calls into question the veracity of any studies that have focused on the educational experiences of neurodivergent students, and as such, establishing the existence or lack thereof a sizable population of students with characteristics of these conditions is of particular interest to our field.

This study seeks to provide initial insight into the relationship between undergraduate computer science students and characteristics of the autism spectrum traits. To that end, this paper explores two research questions:

RQ1. Are some autism spectrum traits prevalent amongst undergraduate students taking introductory computer science classes?

RQ2. If so, does the prevalence of these characteristics differ from that of the general population?

2 Background

Although there are many areas of interest with regard to neurodiversity and the technology landscape, we are particularly concerned with establishing the connection between computing and autism. As such, we establish details on three important topics: (1) ASD, its correlation with careers in computing and technology fields; (2) The data and analyses supporting an increasing incidence of

ASD within the global general population; (3) The Autism Spectrum Quotient (AQ), a widely used, self-administered measure that can aid in determining where an individual may lie on the autism spectrum.

2.1 ASD, Computing Careers, and Familial Multiple-Incidence

One significant consideration supporting the need for further research into neurodiversity within computing education is the growing evidence of a correlation between the prevalence of autism spectrum disorder amongst computing professionals and their families. Several studies have noted a potential relationship between living and working in a computing-heavy region and an increased prevalence of autism spectrum conditions as well as an increased prevalence of autism spectrum conditions within families where the parents themselves present autism spectrum traits [13] [14].

Of particular importance is a prevalence study that was conducted in the Netherlands, herein referred to as *the Eindhoven study* [13]. The study was conducted in the late 2000s across three regions: Eindhoven, Haarlem, and Utrecht. These three regions have a similar demographic and socioeconomic makeup. A key distinguishing feature between Eindhoven, the experimental region, and Haarlem and Utrecht, the control regions, is that 30% of jobs in Eindhoven are part of the computing industry, compared with 16% and 17% of the control regions.

The Eindhoven study specifically collected data on formal, clinical diagnoses of autism spectrum disorder as well as other neurodevelopmental disorders, such as ADHD. 369 schools across the regions, incorporating both traditional schools and those designated as supporting special needs children, participated in the study. They provided data on 62,505 children. In this data, the researchers found that the prevalence of autism spectrum diagnoses were significantly higher in Eindhoven (229 per 10,000) compared to Haarlem and Utrecht (84 and 57 per 10,000 respectively). In other words, where the incidence in the control regions was below 1%, the incidence in Eindhoven was more than 2%.

Additional studies have found similar results in different populations and distributions, such as data collected from 450,395 individuals in the United Kingdom that found that autism spectrum characteristics were more prevalent amongst STEM professionals than the general population [14]. This study used the *Autism Specrum Quotient (AQ)*, discussed below.

2.2 Increasing Incidence and Demographics of ASD

Accompanying the correlations between ASD, computing careers, and familial multiple incidence is the generally increasing incidence of ASD amongst the general population. Some recent studies have shown that 2-3% of individuals,

if not more, may be affected by ASD, up from sub-1% estimations in past decades [16]. Although studies of ASD are orders of magnitude more common in the Western world, more studies are being conducted elsewhere, and they are demonstrating many of the same trends [15] [8].

The professional consensus does not indicate that these changes in incidence rate are caused by overdiagnosis due to updates to the diagnostic criteria of ASD and other related disorders. Wider access to information and healthcare, continuous work to reduce stigmas, and growing research efforts to understand the causes and impacts of ASD are all considered more likely to be the driving forces behind reported increases, especially amongst demographics where the stigmas of ASD, mental disorders, or healthcare in general are traditionally prominent [5] [3].

Furthermore, it is widely understood that biological males are more likey to receive a diagnosis of ASD, with a 4:1 diagnosis ratio compared to biological females. There are several theories on the causes of this difference, from potential genetic markers that are more prominent amongst biological males to perceptual difficulties amongst diagnosticians in identifying ASD traits that may be more effectively masked by females with undiagnosed ASD [2].

2.3 The Autism Spectrum Quotient

The Autism Spectrum Quotient (AQ) is an instrument designed to provide a brief, self-administered measure that can aid in determining where an individual adult, with normal intelligence, lies on the autism spectrum [1].

The AQ is structured as 50 statements with four potential agree or disagree responses; no neutral positions are presented. Responses are given 1 point if the response is indicative of autism spectrum characteristics, or 0 points if the response is indicate of neurotypicality, up to 50 points total. One half of the statements are written such that agreement is indicative, and the other half are written such that disagreement is indicative.

An individual scoring 36 or higher is considered likely to be on the autism spectrum; we refer to this herein as a "clinically significant" score. Outcomes correlating this score with clinical diagnoses have been validated in previous studies. As mentioned, this is a brief and self-administered measure, and as such is not intended to provide a diagnosis of any kind.

3 Methodology

Our survey comprises three sections: (1) demographic questions, (2) program comprehension questions, and (3) the Autism Spectrum Quotient.

Our demographic questions focused on the participaints sense of personal identity and their parental educational and professional history. Specifically,

we sought to document the ethnic and gender identities of the participants, and whether or not either of the parental figures of the participants currently or formerly work in a computing field. This specificity is potentially relevant to the aforementioned broader body of research and understanding of the autism spectrum.

The second section of our survey consists of validated program comprehension questions [4], designed to measure ability in introductory programming tasks. This was included so that our survey was not exclusively behavioural in nature. The third section of the survey was the Autism Spectrum Quotient, previously described.

Our survey was distributed to 10 class sections across 3 courses, covering computer science courses that introduce students to programming concepts and computational theory. These courses are taken by a variety of majors, with some sections providing rudimentary CS instruction to other STEM disciplines, and one course being a required for all majors participating in an interdisciplinary honors program.

Our study was approved by our Institutional Review Board and informed consent was obtained from all participants. ASD can only be diagnosed by a licensed medical professional in a clinical, diagnostic setting, and as such no information regarding clinical outcomes was provided.

4 Results

From the 10 sections that we distributed our survey to, we received 244 responses, of which 169 were valid. Normality of the valid data was evaluated with Shapiro-Wilk, Anderson-Darling, and chi-square tests. We found the null hypothesis rejected in each case ($p < 0.01$, $p = 0.05$, and $p < 0.01$ respectively), indicating that it is likely that our data is normally distributed. Each of the valid 169 responses had their AQ scores computed. Figure 1 shows the distribution of the valid responses. Of the 169 responses, 9 (5.33%) resulted in clinically significant AQ scores.

The responses were then labelled with key demographic categories that have been previously correlated to either the likelihood of receiving an autism diagnosis (e.g., ethnicity, parents working in technology) or to socioeconomic status (e.g., highest level of education attained by mother) [10] [6]. Not all participants could be represented in each category, as not all students supplied valid answers to the demographic questions presented in the survey. These demographic categories are presented in table 1, accompanied by the number of participants with significant AQ scores and the percentage of significant scores within each category.

We then compared all 169 responses and the responses associated with each

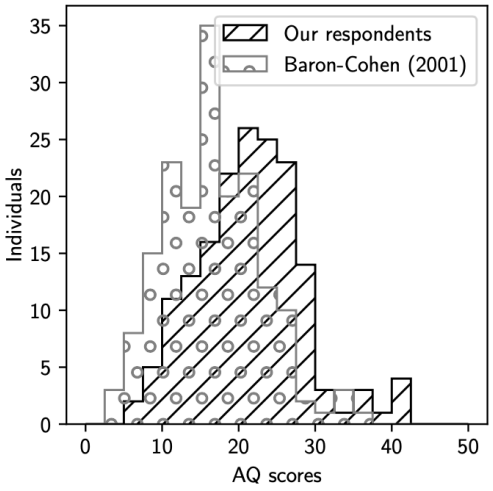


Figure 1: A histogram of participant AQ scores.

Table 1: Clinically significant responses by demographic.

Demographic	Total	AQ $\geq$ 36 (percent)
All valid responses	169	9 (5.33%)
CS-1 students	88	4 (4.55%)
CS-2 students	54	4 (7.41%)
CS-3 students	14	0 (0.00%)
Honors students	13	1 (7.69%)
Male	76	5 (6.58%)
Female	43	4 (9.30%)
Other	1	0 (0.00%)
No gender indicated	49	0 (0.00%)
White	104	9 (8.65%)
Asian	32	1 (3.12%)
Hispanic	16	1 (6.25%)
Mother education less than college	49	0 (0.0%)
Mother education undergraduate	74	5 (6.76%)
Mother education postgraduate	31	4 (12.9%)
Father in computing	24	1 (4.17%)
Mother in computing	16	2 (12.5%)
Both parents in computing	9	1 (11.11%)

label against data provided by a previous validation of the AQ instrument [1]. We compared with the overall control group ($n = 174$, mean AQ = 16.4), male control subgroup ($n = 76$, mean AQ = 18.3), and female control subgroup ($n = 98$, mean AQ = 15.4) reported in said validation using Welch's unequal variances t-test.

There was a statistically significant increase of mean between all valid responses (mean AQ = 21.2) and all three control response groups (all $p < 0.01$; male $p < 0.01$; female $p < 0.01$). There was a significant increase amongst male respondents against controls (mean = 21.4, $p < 0.01$). There was a significant increase amongst female respondents against the overall control group and the female control group (mean = 20.7, $p < 0.01$), but not against the male control group. Of respondents who reported their mother does not have a college degree, there was an increase against the overall control and female control (mean = 20.3, $p < 0.01$), but not the male control. Of respondents who reported their mother has a college degree, there was an increase against all control groups (mean = 20.9, $p = 0.02$ for undergraduate degrees and mean = 23, $p < 0.01$ for postgraduate degrees). When comparing respondents with only one parent who has a computing career, there was an increase against the overall control and female control (mean = 20.2, $p = 0.02$ for fathers and mean = 22.6, $p < 0.01$ for mothers), but not the male control. There were no differences between respondents that reported both parents as having a computing career and the controls.

5 Discussion and Conclusion

The primary aims of this study, elucidated in the introduction, were to discover if some autism spectrum traits are prevalent amongst undergraduate computer science students and to discover if there is a difference in the prevalence of these traits when compared to the general population. Our results strongly support that there is an increased prevalence of autism spectrum traits within our experimental population that significantly exceeds the general population. This correlates with the findings of similar studies of related populations, such as professionals in STEM and the children of technology workers [13] [14].

In all of the identified demographic groups in our study, we saw an increase in the average AQ score of the respondents over that of the general population [1]. This increase in average AQ score was statistically significant in the majority of groups. In cases lacking statistical significance, the common denominator appears to be a representation of the generally higher recorded prevalence of autism spectrum traits amongst male populations. Whilst this trend may exist in our data, we cannot dismiss that this may be a result of comparatively low respondent counts in these cases. This would mean that further data on these

demographics may yield a different analysis, more akin to that of the majority of our results. This is likely to be true, both because we saw a statistically insignificant increase in mean scores in these cases and because there is growing evidence that ASD rates amongst female populations are vastly underreported and underdiagnosed [11].

Our results do face some threats to validity. One issue is in our sample size. The jurisdiction within which our study was conducted considers all inidivudals under the age of 19 to be minors. As some students may be 17 or 18, soliciting their responses was not possible during this study. Additionally, our sample size was constrained by the limited number of participating classes and institutions. The degree of impact these caused is unknown. Finally, there are significant risks involved in requesting medical data or involving clinicians in analyses that may constitute a diagnosis. Having this information, however, would have allowed us to determine if participants do in fact have autism spectrum disorder or a different but often comorbid neurodevelopmental disorder such as ADHD. Without these data, we cannot assert a relationship with clinical diagnoses or rule out interference from other disorders.

In this study, we confirmed the presence of a relationship between higher prevalence of autism spectrum characteristics represented by AQ scores and the pursuit of undergraduate computer science education. This supports previous studies using the AQ instrument and establishes the need for further validation of the relationship. This further prompts for investigation of the efficacy of popular pedagogical methods in computing education and their impacts on our substantial neurodivergent population. We also expect to see validation of this relationship with a greater quantity and range of participants, including students in other fields and stages of their education. Ultimately, we hope that our study will encourage the development of improvements to computing education that will allow all students to succeed in our field.

References

[1] S. Baron-Cohen et al. "The Autism-Spectrum Quotient (AQ): Evidence from Asperger Syndrome/High-Functioning Autism, Males and Females, Scientists and Mathematicians". In: *Journal of Autism and Developmental Disorders* 31.1 (Feb. 2001), pp. 5–17. ISSN: 0162-3257. DOI: 10.1023/a:1005653411471. PMID: 11439754.

[2] Simon Baron-Cohen et al. "Why Are Autism Spectrum Conditions More Prevalent in Males?" In: *PLOS Biology* 9.6 (June 14, 2011), e1001081. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1001081. URL: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001081 (visited on 09/12/2023).

[3] Sander Begeer et al. “Underdiagnosis and Referral Bias of Autism in Ethnic Minorities”. In: *Journal of Autism and Developmental Disorders* 39.1 (Jan. 2009), pp. 142–148. ISSN: 0162-3257. DOI: 10.1007/s10803-008-0611-5. PMID: 18600440.

[4] Ryan Bockmon et al. “(Re)Validating Cognitive Introductory Computing Instruments”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19: The 50th ACM Technical Symposium on Computer Science Education. Minneapolis MN USA: ACM, Feb. 22, 2019, pp. 552–557. ISBN: 978-1-4503-5890-3. DOI: 10.1145/3287324.3287372. URL: https://dl.acm.org/doi/10.1145/3287324.3287372 (visited on 06/16/2023).

[5] M. Catherine DeSoto and Robert T. Hitlan. “Professional Opinion on the Question of Changes in Autism Incidence”. In: *Open Journal of Psychiatry* 03.02 (2013), pp. 61–67. ISSN: 2161-7325, 2161-7333. DOI: 10.4236/ojpsych.2013.32A010. URL: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/ojpsych.2013.32A010 (visited on 09/20/2023).

[6] Jani Erola, Sanni Jalonen, and Hannu Lehti. “Parental Education, Class and Income over Early Life Course and Children’s Achievement”. In: *Research in Social Stratification and Mobility* 44 (June 1, 2016), pp. 33–43. ISSN: 0276-5624. DOI: 10.1016/j.rssm.2016.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0276562416300038 (visited on 07/19/2024).

[7] Carl Haynes-Magyar. “Neurodiverse Programmers and the Accessibility of Parsons Problems: An Exploratory Multiple-Case Study”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Mar. 7, 2024), pp. 491–497. DOI: 10.1145/3626252.3630898. URL: https://dl.acm.org/doi/10.1145/3626252.3630898 (visited on 07/19/2024).

[8] Christina Mohr Jensen, Hans-Christoph Steinhausen, and Marlene Briciet Lauritsen. “Time Trends Over 16 Years in Incidence-Rates of Autism Spectrum Disorders Across the Lifespan Based on Nationwide Danish Register Data”. In: *Journal of Autism and Developmental Disorders* 44.8 (Aug. 1, 2014), pp. 1808–1818. ISSN: 1573-3432. DOI: 10.1007/s10803-014-2053-6. URL: https://doi.org/10.1007/s10803-014-2053-6 (visited on 09/20/2023).

[9] Devorah Kletenik et al. “From Awareness to Action: Teaching Software Accessibility for Neurodiverse Users”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Mar. 7, 2024), pp. 687–693. DOI: 10.1145/3626252.3630859. URL: https://dl.acm.org/doi/10.1145/3626252.3630859 (visited on 07/19/2024).

[10] Sylvia E. Korupp, Harry B. G. Ganzeboom, and Tanja Van Der Lippe. "Do Mothers Matter? A Comparison of Models of the Influence of Mothers' and Fathers' Educational and Occupational Status on Children's Educational Attainment". In: *Quality and Quantity* 36.1 (2002), pp. 17–42. ISSN: 00335177. DOI: 10.1023/A:1014393223522. URL: http://link.springer.com/10.1023/A:1014393223522 (visited on 07/19/2024).

[11] Rachel Loomes, Laura Hull, and William Polmear Locke Mandy. "What Is the Male-to-Female Ratio in Autism Spectrum Disorder? A Systematic Review and Meta-Analysis". In: *Journal of the American Academy of Child & Adolescent Psychiatry* 56.6 (June 1, 2017), pp. 466–474. ISSN: 0890-8567. DOI: 10.1016/j.jaac.2017.03.013. URL: https://www.sciencedirect.com/science/article/pii/S0890856717301521 (visited on 06/15/2023).

[12] Elizabeth Pellicano and Jacquiline den Houting. "Annual Research Review: Shifting from 'Normal Science' to Neurodiversity in Autism Science". In: *Journal of Child Psychology and Psychiatry* 63.4 (2022), pp. 381–396. ISSN: 1469-7610. DOI: 10.1111/jcpp.13534. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jcpp.13534 (visited on 06/15/2023).

[13] Martine T. Roelfsema et al. "Are Autism Spectrum Conditions More Prevalent in an Information-Technology Region? A School-Based Study of Three Regions in the Netherlands". In: *Journal of Autism and Developmental Disorders* 42.5 (May 1, 2012), pp. 734–739. ISSN: 1573-3432. DOI: 10.1007/s10803-011-1302-1. URL: https://doi.org/10.1007/s10803-011-1302-1 (visited on 06/15/2023).

[14] Emily Ruzich et al. "Sex and STEM Occupation Predict Autism-Spectrum Quotient (AQ) Scores in Half a Million People". In: *PLOS ONE* 10.10 (Oct. 21, 2015), e0141229. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0141229. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0141229 (visited on 06/15/2023).

[15] Xiang Sun et al. "Autism Prevalence in China Is Comparable to Western Prevalence". In: *Molecular Autism* 10.1 (Feb. 28, 2019), p. 7. ISSN: 2040-2392. DOI: 10.1186/s13229-018-0246-0. URL: https://doi.org/10.1186/s13229-018-0246-0 (visited on 09/20/2023).

[16] Jinan Zeidan et al. "Global Prevalence of Autism: A Systematic Review Update". In: *Autism Research* 15.5 (2022), pp. 778–790. ISSN: 1939-3806. DOI: 10.1002/aur.2696. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/aur.2696 (visited on 09/20/2023).

The Impact of Specifications-Based Grading on Student Learning Outcomes: A Retrospective Analysis*

J. Walker Orr
Department of Electrical Engineering and Computer Science
George Fox University
Newberg, OR 97132
jorr@georgefox.edu

Abstract

Traditional grading practices often fail to accurately reflect student mastery of course objectives and can create barriers to authentic learning. This retrospective analysis examines the effects of specifications-based grading (SBG) on student learning outcomes in comparison to traditional grading methods in an upper-division computer science course. Drawing from five years of institutional data, we compare student performance metrics across two periods: two years using traditional grading methods followed by three years using specifications-based grading. The study focuses on unit test completion rates as a key metric for evaluating the effectiveness of each grading approach, analyzing 7 programming assignments completed by students over multiple academic years. Results demonstrate significant improvements in assignment completion rates during the first four assignments under specifications-based grading, with increases of 30.4% in unit test completion. However, completion rates for later assignments showed no significant difference between grading approaches, suggesting that students strategically targeted specific grade levels. This research contributes to the growing discussion around

alternative assessment strategies in computer science education and provides empirical evidence for the practical implications of transitioning to specifications-based grading in algorithms courses.

1 Introduction

Assessment practices in computer science education have increasingly come under scrutiny as educators seek methods that better align with learning objectives and professional practice. Traditional point-based grading systems, while familiar, often fail to accurately measure student mastery and can inadvertently encourage grade-seeking behaviors over genuine learning [9]. This has led to growing interest in alternative assessment approaches, particularly specifications-based grading (SBG), which emphasizes mastery demonstration over point accumulation.

Specifications-based grading, also called mastery-based or standards-based grading, represents a fundamental shift in assessment philosophy. Rather than averaging performance across assignments, SBG requires students to demonstrate competency in specific skills or learning objectives through clearly defined criteria [7]. Though there are different frameworks for specifications-based grading, a common approach is to give student assignments pass/fail evaluations based on meeting predetermined specifications, with opportunities for revision and resubmission until mastery is achieved.

The theoretical foundations of specifications-based grading draw from multiple educational frameworks. Bloom's mastery learning theory demonstrates that aptitude predicts learning rate rather than ultimate achievement level, supporting the principle that students should advance when ready rather than according to fixed timelines [1]. Self-Determination Theory provides additional support, as SBG enhances student autonomy through choice in grade targeting, competence through clear standards, and relatedness through improved instructor-student interactions [3].

This study contributes to the research on SBG for computer science by analyzing the effect of SBG on the quality of student work directly. The comparison is on five years of data from an Analysis of Algorithms course, comparing student performance under traditional grading (2019-2020) versus specifications-based grading (2021-2023). We focus specifically on unit test completion rates as an objective measure of student achievement, providing quantitative evidence for specifications-based grading's impact on learning outcomes in computer science education.

2 Related Work

The movement toward alternative grading practices in computer science education has gained substantial momentum over the past decade. Nilson's seminal work established the foundational framework for specifications grading, demonstrating how pass/fail evaluation combined with student choice can restore academic rigor while motivating students and reducing faculty workload [9]. Her approach has been widely adapted across STEM disciplines, with computer science showing particularly strong interest in adoption due to natural alignment with iterative development practices.

2.1 Theoretical Foundations

Multiple learning theories support specifications-based grading approaches. Mastery learning, originally developed by Bloom, provides the core pedagogical framework by emphasizing that learning time varies while learning outcomes should remain constant [1]. This contrasts sharply with traditional grading's fixed time, variable outcome approach. Contemporary research demonstrates that traditional grading practices lack empirical support despite over 100 years of use, while competency-based approaches consistently improve learning outcomes [11, 5].

Self-Determination Theory offers additional theoretical support through its emphasis on autonomy, competence, and relatedness as fundamental psychological needs [3]. Specifications-based grading addresses autonomy through student choice in grade targeting, competence through clear standards and revision opportunities, and relatedness through enhanced feedback interactions. Recent research has documented these psychological benefits in practice, with students reporting increased motivation and engagement under specifications-based systems [4].

2.2 Computer Science Implementations

Computer science courses are particularly well-suited to specifications-based grading implementations. Programming assignments naturally align with pass/-fail evaluation through unit testing frameworks, while iterative development practices mirror the revision opportunities central to specifications grading. Although recent research on the state of SBG for computer science shows that there are a variety of implementation methodologies and limited evaluation of its effectiveness [6].

Recent large-scale implementations have provided compelling evidence for specifications grading's effectiveness. An implementation of SBG in an algorithms courses serving 200-400 students demonstrated successful scaling while

maintaining academic rigor [2]. Their work showed that specifications-based grading could accommodate diverse teaching staff and large enrollments without compromising educational quality.

Tuson and Hickey's implemented SBG in a software engineering course with 142 students and administered the course's quizzes and assignments with the Master Learning App. The effectiveness of SBG was evidenced by student exams and educators also felt that grading was easier despite the need for quite turnaround time [12].

2.3 Assessment and Completion Rate Research

Recent research has increasingly focused on quantitative measures of specifications grading effectiveness. McKnelly and colleagues' large-scale chemistry studies demonstrated improvements in student grades while maintaining learning quality while employing a pass/fail system for assignments [8]. Their research across over 1,000 students provides evidence for improved outcomes even at scale.

Completion rate analysis has emerged as a metric for evaluating alternative grading approaches. Multiple studies document improved assignment completion under specifications-based systems, attributed to students' increased ownership of learning processes [11, 10]. However, systematic measurement approaches remain limited, with most studies using institution-specific metrics that restrict comparative analysis.

Unit testing frameworks provide natural vehicles for objective completion measurement in computer science courses. The integration of test-driven development principles with specifications grading creates synergistic effects where students develop both programming skills and testing competencies through mastery-focused evaluation [10]. This alignment between assessment methods and professional practices represents a key advantage of specifications-based approaches in computer science education.

3 Method

3.1 Course Context

This study analyzes data from a course on the analysis of algorithms, an upper-division computer science course required for the majors. The course covers fundamental algorithmic concepts including graph algorithms, dynamic programming, greedy approaches, and complexity analysis. At our institution, students typically enroll as sophomores, having completed introductory programming sequences and discrete mathematics.

The course remained consistent in content and structure across the five-year study period (2019-2023). Seven programming assignments form the core assessment components: implementation of graph data structures, Dijkstra's algorithm, minimum spanning tree algorithms (Prim's or Kruskal's), 2-Opt algorithm for the Traveling Salesman Problem, Held-Karp algorithm for TSP, genetic algorithm for TSP, and branch-and-bound algorithm for TSP. These assignments were selected to provide comprehensive coverage of major algorithmic paradigms while maintaining consistent complexity levels across years.

3.2 Grading System Comparison

The study compares two distinct grading approaches implemented across consecutive periods. Traditional grading (2019-2020) used weighted point averages across assignments, exams, and participation. Students received partial credit based on rubric criteria, with final grades calculated through standard percentage ranges (90-100% = A, 80-89% = B, etc.).

Specifications-based grading (2021-2023) restructured assessment around pass/fail evaluation of clearly defined criteria. Students selected target grade levels (A, B, or C) corresponding to different completion requirements: A grades required successful completion of all seven assignments, B grades allowed one incomplete assignment, and C grades allowed two incomplete assignments. Failed assignments could be revised and resubmitted unlimited times, generally once per day, before final deadlines.

Pass/fail determination relied primarily on automated unit testing frameworks integrated with manual code quality assessment. Each assignment included comprehensive test suites covering functional requirements, edge cases, and performance constraints. Students received immediate feedback on test results, enabling rapid iteration and improvement cycles.

3.3 Data Collection and Metrics

Data collection focused on unit test completion rates as the primary outcome measure. For each assignment and student, we calculated the percentage of unit tests passed at final submission. This metric provides objective, quantifiable assessment of student achievement independent of grading system variations.

The dataset includes 813 total program submissions across five academic years: 489 submissions from 58 students under traditional grading (2019-2020) and 813 submissions from 82 students under specifications-based grading (2021-2023). All included programs were syntactically correct, submissions that failed to compile were excluded from analysis.

Assignment attempt rates were tracked as a secondary metric, measuring the percentage of enrolled students who submitted each assignment. This pro-

vides insight into student engagement patterns and potential strategic behaviors under different grading systems. Statistical analysis employed two-sample t-tests to compare completion rates between grading approaches for each assignment. Significance testing used $\alpha = 0.05$ throughout.

3.4 Implementation Details

The transition to specifications-based grading required substantial infrastructure development. A fully automated assessment workflow was implemented using GitLab integration, with nightly retrieval of student submissions, automated unit testing, and feedback delivery through Gitlab's issue tracking systems. This automation was essential for managing the increased assessment frequency inherent in revision-based approaches.

Students received detailed instructions on specifications-based grading during the first week of courses using this approach. Clear documentation outlined grade requirements, revision processes, and deadline structures. The feedback system enabled students to monitor their achievement status throughout the semester.

The specifications-based implementation maintained academic rigor through comprehensive unit test suites and detailed code quality requirements. Students were required to demonstrate both functional correctness and professional-level code organization, documentation, and efficiency. This multi-dimensional assessment approach ensured that pass/fail evaluation reflected genuine mastery rather than superficial completion.

4 Results

4.1 Unit Test Completion Rates

Analysis of unit test completion rates reveals substantial differences between grading approaches, with patterns varying significantly across the semester progression. Figure 1 presents completion rates for each assignment under both traditional and specifications-based grading systems.

The first four assignments demonstrate statistically significant improvements under specifications-based grading. Assignment 1 (Graph implementation) improved from 77.6% to 99.8% completion ($p < 0.0001$). Assignment 2 (Dijkstra's algorithm) increased from 62.7% to 94.5% ($p < 0.0001$). Assignment 3 (MST algorithms) rose from 64.5% to 95.1% ($p < 0.0001$). Assignment 4 (2-Opt TSP) improved from 51.7% to 89.0% ($p < 0.0001$).The average improvement across the first four assignments was 30.4%, demonstrating consistent benefits of specifications-based approaches for foundational algorithmic implementations.

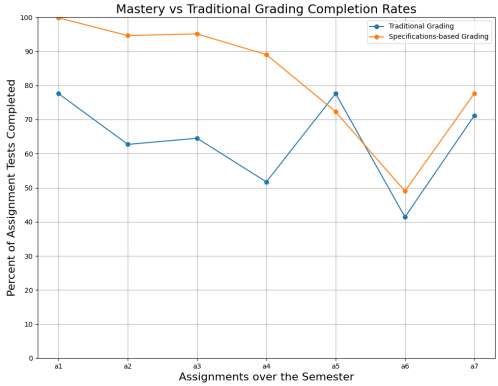


Figure 1: Unit test completion rates by assignment and grading system. Specifications-based grading (orange) shows substantial improvements for assignments 1-4, and little or no difference for assignments 5-7.

Assignments 5-7 show markedly different patterns. Assignment 5 (Held-Karp) completion rates were 77.6% for the traditional grading versus 72.3% under specifications-based grading ($p = 0.4261$). Assignment 6 (Genetic algorithm) showed 41.4% versus 49.0% completion respectively ($p = 0.2683$). Assignment 7 (Branch-and-bound) demonstrated 71.1% versus 77.6% completion ($p = 0.3534$). None of these differences reached statistical significance. Overall for all the assignments, the average improvement was 18.7%.

4.2 Assignment Attempt Rates

Assignment attempt rates provide additional insight into student engagement patterns under different grading systems. Figure 2 shows the percentage of enrolled students who submitted each assignment.

Traditional grading maintained relatively stable attempt rates throughout the semester, ranging from 96% for Assignment 1 to 79% for Assignment 7. The decline was gradual and consistent, reflecting typical student engagement patterns in traditionally graded courses.

Specifications-based grading showed higher initial attempt rates (100% for Assignment 1) but steeper decline for later assignments. Assignment 5 attempt rates dropped to 74%, Assignment 6 to 58%, and Assignment 7 to 79%. This pattern suggests strategic behavior where students targeted specific grade levels and ceased work once requirements were met. That is, completing 5 as-

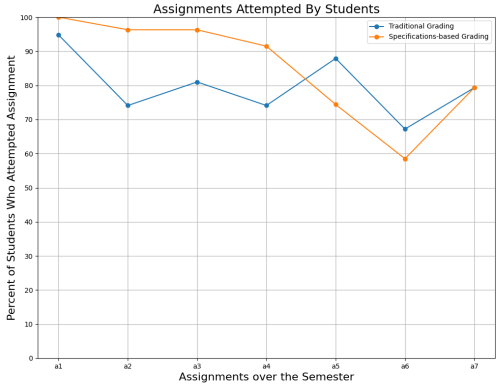


Figure 2: Assignment attempt rates by students for each grading system. The traditional grading system is relatively level while the specifications-based system declines after the fourth assignment.

signments was required to pass the course, so most students chose to do the first 4 and at least 1 of the remaining 3.

The attempt rate data provides mechanistic explanation for the completion rate patterns observed. Students under specifications-based grading appeared to focus intensive effort on early assignments required for course completion, then strategically chose whether to pursue higher grades through additional assignments.

5 Discussion

5.1 Interpretation of Results

The results demonstrate clear benefits of specifications-based grading for early-semester assignments, with substantial improvements in both completion rates and student engagement. The 30.4% average improvement for assignments 1-4 represents meaningful enhancement in student achievement of foundational algorithmic concepts. These improvements align with theoretical predictions and previous empirical findings in specifications-based grading research.

However, the lack of improvement for assignments 5-7 requires careful interpretation. Rather than indicating specifications-based grading's failure, this pattern likely reflects the intended operation of the grading system. Students under specifications-based grading could achieve C grades by completing as-

signments 1-4 along with 1 additional assignment, B grades by completing just 2 more, and A grades only by completing all seven assignments. The attempt rate data supports this interpretation, showing strategic withdrawal from later assignments by students satisfied with lower grades. This is further supported by a higher attempt rate on the final assignment versus assignments 5 and 6, students strategically chose among the final 3 assignments to earn a passing grade.

This strategic behavior represents a feature, not a bug, of specifications-based grading systems. Traditional grading systems encourage students to attempt all assignments regardless of their target achievement level, often resulting in superficial engagement with challenging material. Specifications-based grading allows students to make informed decisions about their effort allocation based on personal goals and circumstances.

The dramatic improvements in early assignments suggest that specifications-based grading particularly benefits foundational skill development. Assignments 1-4 cover essential graph algorithms and data structures that form the basis for more advanced topics. The revision opportunities and mastery focus of specifications-based grading appear especially valuable for ensuring solid understanding of these fundamental concepts.

5.2 Implications for Computer Science Education

These findings have several important implications for computer science educators considering alternative assessment approaches. First, specifications-based grading can substantially improve student mastery of foundational concepts when implemented with appropriate support infrastructure. The automated assessment workflow proved essential for managing the increased evaluation frequency required by revision-based approaches.

Second, the strategic behaviors observed under specifications-based grading may actually benefit student learning by encouraging realistic goal-setting and sustainable effort allocation. Rather than pushing all students toward maximum achievement regardless of cost, specifications-based grading allows for differentiated engagement based on individual circumstances and objectives.

Third, the implementation challenges should not be underestimated. The transition required substantial upfront investment in creating detailed specifications, comprehensive unit tests, and automated workflow systems. However, the long-term benefits in terms of improved learning outcomes and reduced grading burden justify this initial effort.

The research also highlights the importance of careful specification design. Clear, measurable criteria proved essential for successful implementation, while ambiguous requirements led to confusion and disputes. The integration of automated testing with manual code quality assessment provided comprehensive

evaluation while maintaining efficiency.

5.3 Limitations and Future Research

Several limitations constrain the generalizability of these findings. The study focuses on a single course at one institution, limiting external validity. The specific assignments and course structure may not transfer directly to other algorithmic courses or institutional contexts. Additionally, the transition occurred during the COVID-19 pandemic, potentially confounding results with environmental factors affecting student motivation and engagement.

The study's retrospective design precludes controlled comparison between grading approaches within the same semester. Future research should employ randomized controlled designs where feasible, though ethical considerations may limit such approaches in educational settings.

Longitudinal tracking of student outcomes represents an important avenue for future investigation. Understanding whether improved foundational mastery under specifications-based grading translates to enhanced performance in subsequent courses would strengthen the case for adoption. Similarly, career outcome analysis could illuminate long-term benefits of mastery-focused assessment approaches.

The interaction between specifications-based grading and different student populations deserves additional attention. While theoretical arguments suggest these approaches should reduce bias and support diverse learners, empirical verification across demographic groups remains limited. Understanding how specifications-based grading affects different student populations could inform equity-focused implementation strategies.

6 Conclusions

This retrospective analysis provides empirical evidence for specifications-based grading's positive impact on student learning outcomes in upper-division computer science courses. The substantial improvements in unit test completion rates for foundational assignments demonstrate that alternative assessment approaches can meaningfully enhance student mastery of algorithmic concepts.

The strategic behaviors observed in later assignments highlight specifications-based grading's success in promoting realistic goal-setting and sustainable effort allocation. Rather than requiring all students to pursue maximum achievement regardless of personal circumstances, specifications-based grading enables differentiated engagement based on individual objectives and constraints.

The implementation experience demonstrates that specifications-based grading can be successfully scaled to upper-division computer science courses with

appropriate infrastructure support. The automated assessment workflow, comprehensive specifications, and clear progress tracking proved essential for effective implementation.

These findings contribute to the growing evidence base supporting alternative assessment approaches in STEM education. While challenges exist, the substantial improvements in foundational concept mastery suggest that specifications-based grading merits serious consideration by computer science educators seeking to enhance student learning outcomes.

Future research should focus on longitudinal tracking of student outcomes, controlled comparative studies, and investigation of specifications-based grading's equity implications. As the evidence base continues to grow, specifications-based grading appears positioned to become an established, evidence-based practice in computer science education.

The transition from traditional to specifications-based grading represents more than a change in assessment mechanics, it fundamentally alters the relationship between evaluation and learning, promoting mastery-focused behaviors that align with professional practice expectations. This research provides quantitative evidence supporting that transformation while offering practical guidance for successful implementation.

References

[1] Benjamin S Bloom. "Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolinas and Virginia, Topical Papers and Reprints, Number 1." In: *Evaluation comment* 1.2 (1968), n2.

[2] Lijun Chen et al. "Experience report: Standards-based grading at scale in algorithms". In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. 2022, pp. 221–227.

[3] Edward L Deci and Richard M Ryan. "The "what" and "why" of goal pursuits: Human needs and the self-determination of behavior". In: *Psychological Inquiry* 11.4 (2000), pp. 227–268.

[4] Brian C Graves. "Specifications grading to promote student engagement, motivation and learning: Possibilities and cautions". In: *Assessing Writing* 57 (2023), p. 100754.

[5] Thomas R Guskey. "Breaking up the grade". In: *Educational Leadership* 78.1 (2020), pp. 40–46.

[6] Brian Harrington et al. "Specifications and contract grading in computer science education". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 2024, pp. 477–483.

[7] Matthew Johnson. “A Framework for Implementing Mastery Grading in Computer Science Courses”. In: *ACM Inroads* 12.2 (2021), pp. 40–47.

[8] Kate J McKnelly et al. “Specifications grading at scale: Improved letter grades and grading-related interactions in a course with over 1,000 students”. In: *Journal of Chemical Education* 100.9 (2023), pp. 3179–3193.

[9] Linda B Nilson. *Specifications Grading: Restoring Rigor, Motivating Students, and Saving Faculty Time*. Routledge, 2014.

[10] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. “Automated assessment in computer science education: A state-of-the-art review”. In: *ACM Transactions on Computing Education (TOCE)* 22.3 (2022), pp. 1–40.

[11] Mai Yin Tsoi et al. “Variations in implementation of specifications grading in STEM courses”. In: *Georgia Journal of Science* 77.2 (2019), p. 10.

[12] Ella Tuson and Timothy Hickey. “Mastery learning with specs grading for programming courses”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 2023, pp. 1049–1054.

GitDash: A Data-Driven Dashboard for Monitoring Team Progress in Software Engineering Education*

Rahul Bijoor[1] and Kevin Buffardi[2]
Computer Science
California State University, Chico
Chico, California
[1]rbijoor@csuchico.edu
[2]kbuffardi@csuchico.edu

Abstract

This paper introduces GitDash, a data-informed dashboard designed to visualize and assess student engagement in software engineering team projects. By synthesizing GitHub activity data and teams' self-evaluation scores, GitDash provides instructors with an integrated platform to explore contribution patterns, detect imbalances in participation, and interpret team dynamics across collaboration, commitment, and conflict. The system employs Principal Component Analysis (PCA) and K-means clustering to classify teams into behavioral archetypes and enabling targeted instructional support. Through interactive panels focusing on weekly activity, team trends, and individual member insights, GitDash facilitates real-time monitoring and supports equitable, evidence-based assessment practices. The tool demonstrates how combining behavioral and perceptual data can inform pedagogy and foster more transparent evaluation in collaborative software education.

1 Introduction

In collaborative software development courses, ensuring fair and accurate assessment of individual and team contributions is a persistent challenge. Traditional approaches such as peer evaluations and instructor reviews often overlook real-time team dynamics, introducing bias and delaying intervention for underperforming teams [7, 8]. Research underscores these shortcomings and calls for automated tools that track student engagement and deliver timely, actionable feedback. Beyond individual monitoring, there is a critical need to capture team-level insights so that instructors can identify struggling teams, mediate conflicts, and provide targeted support to help teams stay aligned and productive.

To address these challenges, this research introduces *GitDash*, a tool that analyzes team performance and provides instructors with actionable insights. It draws on data from version control systems and surveys to detect patterns in collaboration, contribution, and engagement [1]. This enables proactive instructor interventions and recognition of high-performing teams.

GitDash features an instructor-facing dashboard that offers a comprehensive view of individual and team contributions, both weekly and cumulatively across the semester. By visualizing performance metrics and enabling team comparisons, the dashboard supports fair and informed assessments [3, 5] and is motivated by research that indicates real-time monitoring and automated assessment improve pedagogical interventions and learning outcomes [2, 9]. This paper details the clustering methodology and dashboard design, aligned with its intention to aide instructional strategies in software engineering (and project-based) education.

2 Related Works

Several studies validated the connection between repository activity and peer assessments. In previous work [1] we found that contribution equality, as measured via Git data, positively correlates with peer evaluations. However, we also recognized that quantity of actions like commits alone can be misleading and depending on it for evaluations may incentivize bad practices. Guttmann et al. [5] echoed this, proposing Git Reporter to distinguish substantive work from superficial activity, while Chen et al. [3] designed ProgEdu to improve fairness by identifying free-riders in the teams.

To improve evaluation reliability, tools like CATME [8] and structured peer assessments have gained traction. Loignon et al. [7] showed that Frame-of-Reference training improves the consistency of peer ratings. These studies collectively support GitDash's design, which combines subjective team evalua-

tions with behavioral development data to help facilitate an instructor's ability to recognize and intervene when potential issues in software engineering teams arise.

3 Methods

To develop a dashboard for monitoring individual contributions in software engineering teams, we collected data over ten semesters from an undergraduate Software Engineering course at California State University, Chico. This required course for Computer Science majors and minors has prerequisites in programming, algorithms, and data structures. Over a 15-week term, students work in small teams on a collaborative software project. The course includes two weekly one-hour lectures and two mandatory one-hour lab sessions focused on team meetings and collaboration, offering a structured setting to observe teamwork. The data from the group projects was obtained through team surveys and version control systems to assess individual and group contributions.

The team project accounts for 60% of the overall grade, evaluated on three criteria with the following weightings: (30%) Usefulness, delivering working software and refining it iteratively based on user feedback; (35%) Design, critical analysis of software design with a focus on maintainability; and (35%) Testing, including comprehensive automated test suites. Instructors adjust individual grades to reflect perceived contributions. At semester's end, students completed CATME-based evaluations of their team's *cohesiveness* and *conflicts* [6]. Participation in team surveys earned about 1% credit, but responses themselves did not directly affect grades. Questions were grouped into Collaboration, Commitment, and Conflict, using five-point Likert scales: 1 indicates infrequent conflict (good) or low collaboration/commitment (bad), and 5 indicates frequent conflict (bad), or high collaboration/commitment (good).

Similar to studies employing GitHub data mining efforts [1, 4, 5] we collected meta-data on each team's contributions using a Python script interfacing with the GitHub API. The collected meta-data included timestamps, the acting student's GitHub account, and the specific type of contribution made. Contribution types encompassed commits (code revisions), pull requests (requests for peer review and integration), code reviews (formal evaluations of pull requests), issues (documentation of software requirements), and comments (communications for clarification or discussion). Throughout this paper, we refer to all such instances of student contributions collectively as artifacts. To support reproducibility and encourage further research, we will share a GitHub repository containing the source code for our data collection and dashboard tools.

Artifacts authored by the instructor or by automated GitHub bots (e.g., dependabot) were excluded from the dataset. To analyze the temporal distri-

bution of contributions, the week of occurrence for each artifact was calculated relative to the project's start date. Weekly granularity was selected as the most appropriate temporal resolution for this analysis, as finer-grained measures (e.g., daily) could introduce noise due to external factors, such as individual students' personal schedules. After completing this data refinement, our data set included **19,095 artifacts** from **94 teams** with **465 unique students**.

To classify teams on the basis of their CATME responses, we employed k-means clustering after dimensionality reduction using Principal Component Analysis (PCA) to enhance interpretability and reduce noise. As input features, we first computed the mean values of the collaboration, conflict, and commitment scores in CATME's team reports. These aggregated measures were standardized and then subjected to PCA to extract principal components capturing the majority of variance across teams. The reduced feature set was subsequently used in the k-means algorithm to identify latent team archetypes.

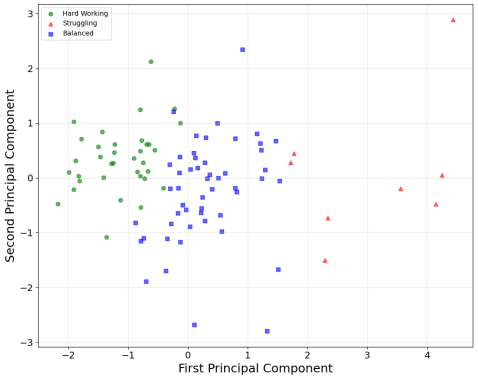


Figure 1: Team Behavioral Archetypes Based on PCA and K-Means Clustering

The resulting clusters are visualized using scatterplot labeled by cluster assignment, which we label as *Hardworking*, *Balanced* and *Struggling* team clusters. This clustering approach enables the identification of teams exhibiting distinct behavioral patterns, thereby assisting instructors in systematically detecting teams that may require additional support or interventions. Figure 1 shows the distribution of teams in the reduced feature space based on the first two principal components of survey data.

This classification of teams identifies their distinct behavioral archetypes. However, instructors have the opportunity to intervene and assist groups if they are provided with insights during the course, so we build a dashboard that integrates the clustering results with GitHub activity data, enabling instructors

to identify teams and individual students in need.

3.1 Dashboard Design and Functionality

We developed an interactive dashboard to support the analysis of team dynamics using both survey responses and GitHub contribution data. The dashboard is organized into three primary panels, each tailored to address specific analytical objectives related to team performance and composition. Users can select filters for team, week, and individual members; by default, the dashboard displays data for all weeks and all members. Also, the dashboard displays the team's computed classification (e.g., "Balanced"), derived from the k-means clustering of survey data, providing a qualitative summary of perceived team dynamics.

The **Weekly Activity panel** of the dashboard is designed to facilitate temporal analysis of team engagement by aggregating contribution data from GitHub repositories on a per-week basis. This panel enables users to examine patterns of activity over time and assess the participation levels of individual team members. As shown in **Figure 2**, the weekly activity is visualized through multiple interactive components.

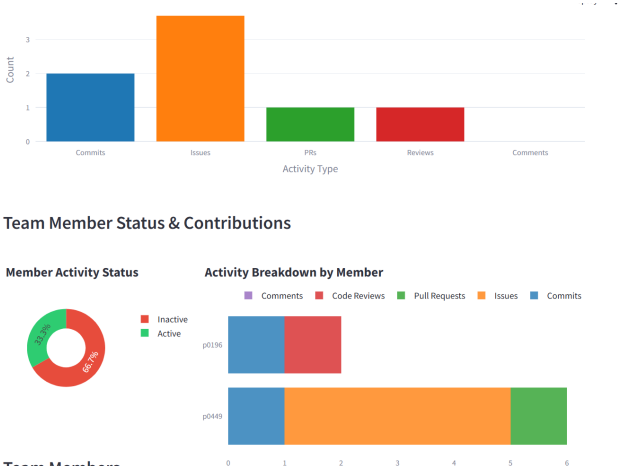


Figure 2: Weekly Contribution Overview: Team and Member Activity Breakdown

A pie chart provides a high-level overview of member engagement, indicating the proportion of team members who were classified as *Active* (i.e., those who made at least one contribution during the selected week) versus *Inactive*. A bar chart presents the breakdown of different activity types such as

commits, issues, pull requests, code reviews, and comments aggregated at the team level for the selected week. A corresponding bar chart details the contribution activity of each team member, enabling comparison of individual effort and involvement. Members are color-coded by status (active or inactive) to visually distinguish participation levels. Together, these visualizations support identification of engagement trends, highlight imbalances in workload distribution, and assist in detecting potential collaboration issues or participation gaps within teams.

The **Team Analysis panel** provides aggregated views of team performance and activity patterns across the project's duration. This panel facilitates comparative analysis between teams and longitudinal tracking of a single team's dynamics. It incorporates several components to summarize team-level data: aggregate contribution metrics display the cumulative counts for commits, issues, pull requests, code reviews, and comments, offering a quantitative measure of overall output. Temporal activity patterns are visualized via a line

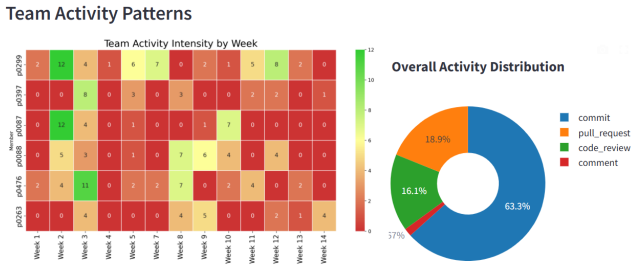


Figure 3: Longitudinal Team Performance: Trends, Heatmaps, and Workflow Patterns

graph illustrating weekly trends for each contribution type, enabling the identification of peak activity periods and shifts in development focus. As shown in **Figure 3**, a heatmap renders team activity intensity, mapping individual member contributions on a week-by-week basis to allow for rapid assessment of engagement consistency and distribution within the team. In addition, a pie chart delineates the proportional breakdown of different contribution types relative to the team's total actions, thereby offering insights into the team's predominant workflow characteristics.

The **Member Insights panel** enables comparative overview of each team members as well as detailed examination of a specific individual's contributions. When "All Members" is selected, the panel, titled "All Team Members Comparison", facilitates assessment of relative participation and consistency

across the team.

It features a stacked bar chart comparing the total volume and type of contributions for each member side-by-side. Additionally as shown in **Figure 4** below, a "Contribution Volume vs Consistency" scatter plot visualizes each member based on their total contributions and their activity consistency (measured as the percentage of weeks with at least one contribution) to help identify engagement patterns, such as contrasting high volume as a result of consistent contributions with those who seldom contribute and do so in large chunks.

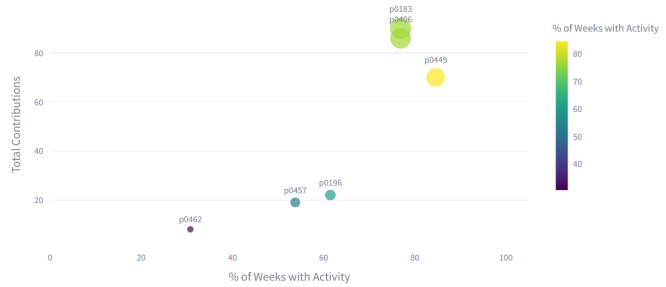


Figure 4: Member Comparison : Total Contributions vs Consistency

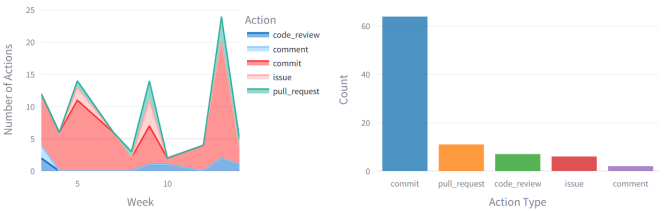


Figure 5: Individual Contribution View: Timeline and Action Breakdown

Alternatively, when a specific student is selected from the filter, the panel focuses on their individual contributions. This granular perspective is essential for assessing individual participation and understanding specific work patterns.

As shown in **Figure 5**, The panel integrates several features for this individual view: contribution summaries provide total counts for each action type attributed to the selected student; a contribution timeline visualizes the frequency and distribution of the student's actions over the project weeks;

an action breakdown bar chart illustrates the composition of the individual's contributions by type; and access to a detailed activity log presents a chronological, time-stamped record of the student's actions, complete with associated metadata such as commit messages or review statuses. This detailed individual view supports nuanced evaluation of performance and contribution styles within collaborative projects.

4 Results

The K-means clustering algorithm, applied after dimensionality reduction using Principal Component Analysis (PCA), yielded three distinct clusters of teams based on their survey responses: *Hardworking*, *Balanced*, and *Struggling*. Table 1 summarizes the mean and standard deviation for each cluster across the three measured dimensions: Commitment, Collaboration, and Conflict.

Cluster	Mean			Standard Deviation		
	Commitment	Collaboration	Conflict	Commitment	Collaboration	Conflict
Hardworking	3.5481	4.3710	1.3227	0.1497	0.3129	0.1808
Balanced	3.1519	3.9260	1.4896	0.2232	0.4021	0.2524
Struggling	2.8726	3.1478	2.2861	0.2957	0.4235	0.5247

Table 1: Cluster-wise Mean and Standard Deviation of Team Dynamics

As shown in the results, the *Struggling* team reported consistently poorer commitment, collaboration, and conflict. These results provide empirical support for the clustering approach, allowing instructors to differentiate teams based on behavioral patterns and survey-reported dynamics. The team classifications generated through this method were integrated into the dashboard, enabling targeted visual analysis of contribution trends and member participation levels.

5 Discussion

The clustering analysis shows that student teams can be effectively grouped based on self-reported collaboration, commitment, and conflict. The distinct *Hardworking*, *Balanced*, and *Struggling* clusters indicate that combining survey metrics with PCA and K-means produces interpretable groupings aligned with team behaviors.

Hardworking teams reported high collaboration and commitment with minimal conflict, reflecting strong cohesion. In contrast, *Struggling* teams showed low collaboration and commitment with elevated conflict, suggesting a need for instructor support. *Balanced* teams exhibited moderate engagement and stable dynamics.

These results highlight the value of combining surveys with clustering to uncover actionable insights. Embedding the classifications into an instructor-facing dashboard enhances real-time monitoring and supports informed pedagogical decisions.

Threats to Validity

While the clustering approach provides a systematic method to differentiate team experiences, several limitations must be acknowledged. Firstly, the classification is based solely on survey responses, which may be subject to self-reporting biases. Students may under-report conflict or overstate collaboration due to social desirability or misunderstanding of survey questions. In addition, the dashboard has been designed as motivated by our Principal Component Analysis, but has not yet been integrated into SE courses. Future work will investigate whether the dashboard enables instructors to identify struggling teams and individuals, and how their interventions impact team outcomes.

6 Conclusion

This study presented GitDash, a data-driven dashboard that enables instructors to monitor individual and team progress in collaborative software engineering courses. By combining GitHub contribution data and CATME-based survey responses, the system applies Principal Component Analysis and K-means clustering to classify student teams into meaningful categories. The resulting clusters of *Hardworking*, *Balanced*, and *Struggling* teams highlight team archetypes and the need for a framework for early identification of teams requiring additional support. The dashboard offers multi-level insights into weekly activity trends, team performance, and individual engagement, facilitating a comprehensive understanding of classroom dynamics in real-time. Future work may explore additional behavioral signals and educational interventions to help struggling teams and individuals adopt healthier teamwork.

References

[1] Kevin Buffardi. "Assessing Individual Contributions to Software Engineering Projects with Git Logs and User Stories". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 650–656. ISBN: 9781450367936. DOI: 10.1145/3328778.3366948. URL: https://doi.org/10.1145/3328778.3366948.

[2] Madison W. Call, Erik Fox, and Gina Sprint. "Gamifying Software Engineering Tools to Motivate Computer Science Students to Start and Finish Programming Assignments Earlier". In: *IEEE Transactions on Education* 64.4 (2021), pp. 423–431. DOI: 10.1109/TE.2021.3069945.

[3] Hsi-Min Chen, Bao-An Nguyen, and Chyi-Ren Dow. "Code-quality evaluation scheme for assessment of student contributions to programming projects". In: *Journal of Systems and Software* 188 (2022), p. 111273. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111273. URL: https://www.sciencedirect.com/science/article/pii/S0164121222000358.

[4] Jian Chen et al. "Assessing Teamwork Performance in Software Engineering Education: A Case in a Software Engineering Undergraduate Course". In: *2011 18th Asia-Pacific Software Engineering Conference*. 2011, pp. 17–24. DOI: 10.1109/APSEC.2011.50.

[5] Michael Guttmann, Aleksandar Karaka, and Denis Helic. "Attribution of Work in Programming Teams with Git Reporter". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 436–442. ISBN: 9798400704239. DOI: 10.1145/3626252.3630785. URL: https://doi.org/10.1145/3626252.3630785.

[6] Andrew C. Loignon et al. "Elaborating on Team-Member Disagreement: Examining Patterned Dispersion in Team-Level Constructs". In: *Group & Organization Management* 44.1 (2019), pp. 165–210. DOI: 10.1177/1059601118776750. URL: https://doi.org/10.1177/1059601118776750.

[7] Andrew C. Loignon et al. "Facilitating Peer Evaluation in Team Contexts: The Impact of Frame-of-Reference Rater Training". In: *Academy of Management Learning & Education* 16.4 (2017), pp. 562–578. DOI: 10.5465/amle.2016.0163. URL: https://doi.org/10.5465/amle.2016.0163.

[8] Misty L. Loughry, Matthew W. Ohland, and D. DeWayne Moore. "Development of a Theory-Based Assessment of Team Member Effectiveness". In: *Educational and Psychological Measurement* 67.3 (2007), pp. 505–524. DOI: 10.1177/0013164406292085. URL: https://doi.org/10.1177/0013164406292085.

[9] Anju Mehta, Nikhil Mehta, and Ishaan Bindal. "Maximizing integrative learning in software development teams: A systematic review of key drivers and future research agenda". In: *Journal of Systems and Software* 190 (2022), p. 111345. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111345. URL: https://www.sciencedirect.com/science/article/pii/S0164121222000838.

Mob Programming in a Software Design Course*

Ben Kovitz
Department of Computer Science
California State Polytechnic University, Humboldt
Arcata, CA 95521
blk14@humboldt.edu

Abstract

Case study of a Software Design course taught using mob programming. Mob programming (also "team programming", "ensemble programming") is collaborative programming in which the whole team—in this case, the whole class—writes code together at the same time, like pair programming but including everyone. The course combined daily collaborative design and coding with intensive daily readings and oral presentations. Students gained most strongly in skills that are normally hard to teach in colleges: egoless cooperation and "tacit knowledge" of software design.

1 Introduction

I taught a course on software design in the Spring 2025 semester at California State Polytechnic University, Humboldt (course number CS 356), predominantly by a new pedagogy: mob programming. Mob programming, also called team programming or ensemble programming, is "a software development approach where the whole team works together on the same thing, in the same space, and at the same computer." [6] Mob programming is like pair programming, but not limited to two programmers at one computer.

Several reasons suggested mob programming as an effective way to teach software design:

1. Experience from industry suggests that pair programming and mob programming result in knowledge spreading rapidly throughout the team. So why not exploit this happy property of mob programming as a class's principal method of learning?

2. Many industry complaints about skills lacking in recent college graduates revolve around "habitability"—the aspects of design that go beyond correctness and relate to programmers' ability to work with code over many years, such as readability, maintainability, testability, and debuggability [2]. The usual college-course format of lectures, homework assignments, and exams often gives students experience only with writing small, standalone pieces of code that just barely work or partly work (enough to get partial credit) and that no one looks at ever again. Could mob programming on one project, building it for most of one semester by piecemeal growth, give students valuable experience working on each other's code, seeing for themselves how to grow a software design that's habitable for others?

3. Software engineering is a complex and collaborative art. Engineers must constantly learn and constantly adjust to unexpected problems and to other people on the development team. Much of the skill is difficult or impossible to express precisely in verbal representations and procedures, and can be acquired only through real practice. This makes it a good fit for the research by anthropologists Jean Lave and Etienne Wenger on learning in "communities of practice" [4]. Could mob programming create an artificial community of practice for one semester?

4. Only seven students were enrolled in the class. CS 356 is a junior-level course, and these students were among the first cohorts in the university's recently begun Software Engineering degree program. So, this semester provided a good opportunity to experiment with mob programming.

Teaching by mob programming could be considered a species of both studio-based learning and flipped classrooms: all have few lectures and much in-class supervised active work. A difference is that in mob programming, all the students, and even the teacher, work on the same problem or project at the same time. Another difference is that rather than just critiquing a student's work, the next "navigator" (see below) simply modifies the code while discussing the change with the rest of the class.

The rest of this article describes how the course went: its structure, problems that arose, solutions that we found or didn't find, and a final assessment.

2 The Semester

2.1 Semester Overview

The semester plan was as follows. (The complete syllabus is available at [3].)

- Each class session was a lab—no lectures, except for an introductory lecture on the first day and occasional improvised mini-lectures as needed. Sessions were one hour and 50 minutes long, twice a week.
- Readings were assigned at the end of each class session, and students had to give brief presentations on the readings at the start of the next session. These presentations usually took up the first 20 to 30 minutes of class time. The readings came only from sources influential among professionals, such as books by Martin Fowler and Scott Meyers, as well as famous blog posts and journal articles, not from textbooks.
- Students were required to keep notebooks in which they recorded the ideas or techniques they learned in each class session, and to periodically hand these in during the semester.
- The first few class sessions: design exercises, mostly omitting implementation, to teach UML and design as something distinct from code.
- Most of the semester (about 10 weeks) was spent designing and implementing a single piece of software in C++, collaboratively by mob programming together in class.
- Last two weeks: we returned to design exercises without implementation, now introducing distributed system design (load balancers, microservices, etc.).
- Last day of class: each student gave a presentation on what they learned during the semester.

2.2 Initial Design Exercises

The first design exercise was borrowed from a job-interview question, involving designing a state-transition diagram for a controller in a common home appliance (details omitted so I can reuse it). The students did this in about an hour of class time, after which I corrected errors and pointed out that they had worked out a mathematically complete description of the software without writing a line of code.

Then we moved on to implementing the finite state machine in C++. Had this been an ordinary class, I would have written the code at a projector,

explaining my thoughts as I went. Instead, we had our first experience with mob programming, which proved much more educational. We set up our mob "rotation" as follows:

- One person, called the "driver", sits at the keyboard of the computer hooked up to a projector.
- One person, called the "navigator", tells the driver what to do next.
- Everyone else ("the mob") watches, checks the code for errors, sees what ideas come to mind, and does anything else they think might be helpful, such as adding to-do items to the whiteboard or searching the Internet for documentation.
- We followed Llewelyn Falco's "strong-style pairing" [1], which has this rule: "For an idea to go from your head into the computer, it must go through someone else's hands." The driver does not enter his or her own ideas into the code. The driver's job is to follow the navigator's instructions. The navigator tries to state each instruction at the highest level of abstraction that the driver can implement. That may take the form of "Now write a test for the null case" or, if necessary, specific keystrokes to help the driver operate an unfamiliar text editor.
- We set a timer so that every 5 minutes, we rotated roles. Someone from the mob becomes the driver, the driver becomes the navigator, and the navigator returns to the mob.
- All the students' names were written in sequence ("the rotation") on the whiteboard, so we always knew who was to drive next. I placed myself eighth (last) in the rotation so that every student would have a chance to navigate before I spoke up.

We drew the state-transition diagram on the whiteboard, and then the students proceeded to do exactly what I think you should not do: they started putting together tricky nested loops with tricky conditions inside—essentially neglecting the state-transition diagram and trying to reproduce the desired output by cobbling together familiar programming constructs in the usual ways.

At one point, the navigator suggested "reversing" a variable for a sensor's input. The driver didn't know how to do that in C++. The navigator said, "Put an exclamation point" (the Boolean negation operator). The driver was surprised by how easy that was, got a brief explanation, and then went on to the next thing. Similar small points came up during mobbing nearly every day. Students came to the class with widely varying prior experience. Some might be lost in a lecture that assumes knowledge they don't have and get further

behind as the semester continues. Mob programming effortlessly paused to fill in just the bit of knowledge that a student needed, right when he or she needed it.

Eventually the students hit on the idea of a variable for the state of the state machine. From the mob, I suggested a `switch` statement, but they still had no systematic approach and were writing extremely bug-prone code. Finally, as the navigator, I had the driver make a `case` for the `switch` statement that fully handled the initial state. The students had "aha!" moments and, as they rotated into the navigator role one by one, worked out how to systematically cover all of the infinite set of possible input sequences. On our first attempt to compile the code, it was correct and passed all tests (which the students were now able to design systematically, too). The students had their first experience with *design* distinct from code, and with a systematic mapping of design elements to program elements.

Some students found the initial mob programming stressful. They'd written almost all of their previous code solo. Now they had to write code in front of a whole class—while thinking it up, unsure of their approach, gaps in their knowledge and skills on display. "Am I doing it right?" some asked me after class, unsure, since I gave them no rubric for "correct" mob programming.

A small problem with the physical environment became apparent: the tables in the lab room were far apart and perpendicular to the projection screen, so it was awkward to look at the screen, and background noise made it hard to be heard across the room. We eventually found a quiet room where the seats faced the projection screen and a new driver could swap into place in a few seconds. Mobbing can be done hovering around a laptop, but these conveniences help.

After that, we moved on to designing an ATM machine, following the method of making classes from nouns in the requirements and assigning responsibilities based on relationships, as explained in [5]. We liked the mob-rotation format so much, we used it for this UML-only, whiteboard-only design process. There was no navigator role, only a driver role that rotated every 10 minutes. The driver wrote on the whiteboard while everyone else suggested items to add or change. Since I was the only one who knew the process, I facilitated.

2.3 The Project

Before the semester, I had asked around campus for software that people needed written and even received some suggestions on LinkedIn. In class, we rated all the options at the whiteboard for feasibility, opportunity to learn design patterns, and suitability for C++, and chose to make a diagram editor. We kept a clear boundary between exploration ("divergent thinking") and evaluation or filtering of ideas ("convergent thinking"), doing first one and then the other.

We spent one class session brainstorming for features. This gave the students experience in collaborative software design at the level of requirements or product definition. The first idea was "little boxes in a canvas". Exploration continued through "TiKZ editor" and, most ambitiously, "the VSCode of diagram editors, allowing plug-ins for electrical simulation of schematic diagrams or anything else". We pared that down to "enough for Ben to draw graphs for homework problems in graph theory, with a *vim*-like user interface."

Next we had to make a crucial design decision: which GUI library to use? I assigned readings on two popular and relatively simple libraries, FLTK and SDL, intending that the students would choose one of those. One student independently researched Qt, a much more complex library—which I had intended to avoid. After presentations on all of them and another group discussion, we chose Qt, attracted by its ability to make a sophisticated GUI in C++ very quickly and run it on both desktop and mobile platforms, deeming this to outweigh the risks of greater complexity and time to learn.

I assigned readings in the Qt documentation and we got back to mob programming. Qt did turn out to be problematic. We spent a lot of time in and out of class getting the Qt libraries to install and work with VSCode on the students' various computers and operating systems. This was chaotic and frustrating—but so is real-life software development, often for reasons just like this. Qt did enable us to quickly and easily try out out radical new user-interface ideas, something we likely could not have done with the other libraries.

We found that mob programming can work at amazing speed—even as it feels like you're working slowly, doing just a tiny bit before passing the baton to the next person. One day, for example, after we implemented snap-to-grid as the user moved an object, we had five minutes left, and our next item was to make the grid visible—too little time, I thought. But the next navigator/driver team dove in, and with a little help from the mob got Qt displaying a visible grid before class was done.

We experimented with the rotation timer and found that we liked a 7-minute rotation best. That was long enough to let the navigator and driver settle in and still allowed each student to get two rounds in during the class period.

One day, a student mentioned that he had no idea how to implement Undo. I assigned readings on the Command and Memento patterns, and the next day, after oral presentations, the class weighed them at the whiteboard and chose Memento because it seemed simpler to implement: just store a vector of snapshots of the current state of the diagram. By the next class session, we found the flaw: coordinating ownership of Qt's diagram objects was unmanageable. It emerged that the students had never fully grasped that in C++, unlike in Java or Python, you distinguish between holding a reference to an object and

holding the object—and the pitfalls of having to bear this responsibility. We deleted the code and reimplemented using the Command pattern—and found not only that it worked correctly, it was actually simpler.

This experience—completely unplanned—implicitly taught a fact about software design that is easy to say but hard to believe until you've experienced it: it's often difficult to tell whether a design idea will be good or bad until after you've implemented it. In hindsight, it's easy to say that we "should" have chosen Command on the first day, but of course that's only in hindsight.

2.4 Testing the edges of mob programming

The Memento/Command revision prompted us to stray from strict mob programming and stop the timer for about 45 minutes—the time needed to explain and reach consensus about the problem, including understanding that Qt was written long before C++11's `unique_ptr` and `shared_ptr` and follows different conventions for ownership of dynamic memory.

The navigator role turned out to induce the most stress. As driver, you're sitting at the keyboard typing code in front of everyone, but as navigator, you're standing in front of the everyone and you bear the responsibility of directing the work. Sometimes a student didn't have ideas yet, or his or her ideas weren't yet clear enough to tell someone else how to implement them. Our mobbers tended to call out ideas while the navigator was still gathering his or her thoughts. Our solution: When you're the navigator in this position, ask, "May I have a moment of silence, please?"

We found that some design ideas required more thought, or a different kind of thought, than happens during mob programming. Sometimes, just getting an idea across takes longer than one rotation. So, on some days, I assigned to one person not a reading, but a design problem and a first implementation. In the next class session, that person gave an oral presentation on what they made, and then the class as a whole reimplemented it from scratch. This produced a result that was often simpler and was of course understood by the whole class, since every student literally had their hands in writing it.

2.5 AI-Assisted Programming

The class's AI policy was to use AI mainly for documentation look-up and simple code-completion with Copilot during the first half of the semester, and during the second half, to run as wild as we could with AI and see all that it can do. Right after Spring Break, a student suggested writing a short document describing all the features we wanted and letting ChatGPT rewrite the entire diagram editor from scratch.

So, we mob-wrote a user-interface design document in a wiki page and let ChatGPT rip. Its code did not compile, nor did revising the document to tell it not to make those errors again fix it. We manually fixed the problems and made some rapid progress.

But then the class rebelled! The students found that the fast cycle time was so fast, they did not understand the code that ChatGPT was producing. They wanted to understand the programming and design ideas, not merely see them rushing by. So we reverted to our AI policy from the first half of the semester, and actually used Copilot more sparingly.

3 Assessment and Conclusions

3.1 Results

The students reported two main things that they learned from the class:

1. How to collaborate. The students had all done some pair programming in earlier classes, but they had never seen collaboration at this intensity. I had meant for mob programming to give the students broad practical experience with software design, and it did, but collaboration itself took the starring role.

2. The parts of software development that are too hard to put into words. For example, how much forethought should you put into code before writing it? How quickly does refactoring lead a good design to emerge "spontaneously" and under what circumstances does this happen or fail to happen? How much design is overdesign?

The students reported that they got extraordinary practice in oral communication, partly from the daily oral presentations and partly from the discipline of verbalizing every design decision that mob programming imposes. Mob rotation circumvented the common problem of vocal students dominating a class and quiet students not asking about their concerns.

The biggest disappointment of the course was that 10 weeks was not enough to implement a satisfactory "graph editor for homework assignments". Ten weeks sounds like a long time, but meeting twice a week, that amounts to only about 30 hours of mob programming. That's less than a week of work at a regular job, as the students were learning many new and unfamiliar concepts.

Another disappointment was that the students got only a little exposure to unit tests and test-driven design. The students all made a test or two in GoogleTest, but we never used it in the diagram editor because unit-testing a GUI is hard. I would recommend in future mob-programming courses to

implement a nontrivial back-end for a web site with a simple interface, or simply a program that has only a textual user interface.

The course was in some ways easy to teach and in other ways hard to teach. The students did most of the lecturing, not me. However, the course also required me to pay close attention and adjust continuously and creatively throughout the semester, more than in other classes. While I had selected a set of readings before the semester, I had to choose and search for readings before each class to suit whatever the programming was leading us to next.

The syllabus listed 78 topics of which I hoped to cover "some large subset". I count about 55 of them touched on or explored to varying degrees: the DRY principle, *git* and version control, refactoring, code smells, information-hiding, Model–View–Controller and other design patterns, adding a level of indirection, writing the calling code first, YAGNI, coupling and cohesion, and many more. Most of these topics came up naturally, without lectures, as they arose in the course of writing code and talking about it; some needed readings.

The course's focus on tacit knowledge gained through experience made designing the final exam tricky. Nevertheless there was much explicit knowledge that could be tested: designing a finite-state machine to solve a problem, making UML diagrams given requirements for a simple distributed system, articulating a reason for an opinion about software design. The final exam demonstrated weakness in the students' understanding: most answers to the "hard" problems had some serious flaw. Perhaps that's to be expected in a course with more breadth than depth, but I would recommend in future mob-programming courses that a homework assignment to be done solo should follow each central topic, even if this means going a day without new readings.

Course evaluations averaged 4.8 on a scale of 0 to 5, the highest I've received as a teacher. I am most pleased that the students *know* what they learned in the course. Their knowledge is imperfect, but they know it first-hand, not "because I was told that in college."

3.2 Risks and Unknowns

Three important unknowns are unaddressed by this experience. First, what if a student is uncooperative? An occasional problem in industry, more common in people of college age, is the intransigent "prima donna"—the opinionated programmer who knows better than anyone else, sees faster than anyone else what to do, and sees no value in patiently working to build consensus.

Second, what if a student is unable to keep up? Programming is hard, especially at the nearly professional level reached in this course, and not all students have the talent or prior knowledge needed to do it. Some students hold back and let their teammates do most of the work—a common problem in group projects, even at the senior level.

In ordinary classes, a few unpleasant or lazy students don't bring down the class. But in an intensely collaborative class, they could spoil the experience for all. I was fortunate to have a class filled completely with cheerful, dedicated, cooperative students; not all mob-programming classes may be so lucky.

And third, could the mob-programming approach be brought to a larger class? A mob of eight is already quite large, requiring each student to wait a long time between turns as driver; a mob of 30 would be impossible. One idea is to have multiple mobs of about 4 students, with the teacher and/or students "floating" between different mobs, carrying information along and being brought up to speed upon joining each new mob. Would this require multiple projectors? If so, the expense might be prohibitive.

4 Acknowledgements

Much gratitude to the many people who helped make the course a success: Shahzad Aslam-Mir for wide-ranging advice on everything from projects to tools to C++ libraries; Rebecca Wirfs-Brock for inviting me to look into mob programming and guidance finding the right balance between giving too much and too little direction to the students; Woody Zuill for generously making time to answer my many basic questions about mobbing; Austin Chadwick and Chris Lucian for generously making time to answer yet more questions about mobbing and letting me mob remotely at Hunter Industries; Dave Bender for the initial design exercise; Sherrene Bogle for creating the original CS 356 course, making this experiment possible; and most of all to the students, who dove boldly, cooperatively, and creatively into a wildly experimental course.

References

[1] Llewellyn Falco. *Llewellyn's strong-style pairing.* Blog post. June 2014. URL: `http://llewellynfalco.blogspot.com/2014/06/llewellyns-strong-style-pairing.html`.

[2] Richard P. Gabriel. *Patterns of Software: Tales from the Software Community.* New York: Oxford University Press, 1996. ISBN: 9780195102697.

[3] Ben Kovitz. *CS 356, Software Design, Spring 2025: Course Syllabus.* 2025. URL: `https://github.com/bkovitz/cs356-sp25/blob/master/syllabus/syllabus.pdf` (visited on 07/04/2025).

[4] Etienne Wenger-Trayner and Beverly Wenger-Trayner. *Introduction to communities of practice: A brief overview of the concept and its uses.* 2015. URL: `https://www.wenger-trayner.com/introduction-to-communities-of-practice/`.

[5] Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. *Designing Object-Oriented Software*. Englewood Cliffs, NJ: Prentice Hall, 1990. ISBN: 0136298257.

[6] Woody Zuill and Kevin Meadows. *Software Teaming: A Mob Programming, Whole-Team Approach*. 2nd. Independently published, 2022. ISBN: 9798361186938.

Reviewers — 2025 CCSC Northwestern Conference

Lucas Cordova ..... Willamette University, Salem, OR
Henry Walker ..... Grinnell College, Grinnell, IA
Jeremy Thompson ..... Washington State University, Everett, WA
Joseph Skudlarek ..... Lewis & Clark College, Portland, OR
Heather Kitada Smalley ..... Willamette University, Salem, OR
Alain Kägi ..... Lewis & Clark College, Portland, OR
Paul Bonamy ..... Washington State University, Pullman, WA
Zhiju Yang ..... Seattle University, Seattle, WA
Richard Weiss ..... The Evergreen State College, Olympia, WA
Ben Tribelhorn ..... University of Portland, Portland, OR
Radana Dvorak ..... Saint Martin's University, Lacey, WA
Peter Drake ..... Lewis & Clark College, Portland, OR
Rachel Brown ..... Willamette University, Salem, OR
Fred Agbo ..... Willamette University, Salem, OR
Tammy VanDeGrift ..... University of Portland, Portland, OR
Anna Ritz ..... Reed College, Portland, OR
Anonymous Reviewers .....