The Journal of Computing Sciences in Colleges

Papers of the 34th Annual CCSC South Central Conference

March 31st, 2023
Stephen F. Austin State University
Nacogdoches, TX

Bin Peng, Associate Editor
Park University

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

Volume 38, Number 7 **April 2023**

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 **5**

CCSC National Partners **7**

Welcome to the 2023 CCSC South Central Conference **8**

Regional Committees — 2023 CCSC South Central Region **9**

Reviewers — 2023 CCSC South Central Conference **10**

Keynote Speaker **11**
Paul R. Gault

Supporting Low-Income, Talented Undergraduate Students in Engineering and Computing Sciences with Scholarships and Mentoring **12**
Dulal C. Kar, Scott A. King, and Dugan Um, Texas A&M University-Corpus Christi

Iterative Efforts for Improving Learning Experience in Software Engineering **27**
Pradip Peter Dey, Mohammad Amin, Bhaskar Raj Sinha, National University

Bug Battles: A Competition to Catch Bugs in Different Programming Languages **36**
Waleed Alhumud, Abdullah Alenzi, Renée Bryce, Yuan Li, University of North Texas; Nasser Alshammari, Jouf University

Preparing ABET Accreditation for An Undergraduate Software Engineering Program **46**
Jicheng Fu, Myungah Park, Gang Qian, Hong Sung, Thomas Turner, University of Central Oklahoma

Comparative Sequential and Parallel Discrete Signal Convolution Algorithms: A Case Study **55**
Caleb Sneath, Eduardo Colmenares, Midwestern State University

Developing Incident Response-Focused Cybersecurity Undergraduate Curricula **65**
Junghwan "John" Rhee, Myungah Park, Fei Zuo, Shuai Zhang, Gang Qian, Goutam Mylavarapu, Hong Sung, Thomas Turner, University of Central Oklahoma

Hands-On Lab Development for Policy Violations in Voice Personal Assistants **75**
Alejandra Enriquez Sanchez, Oludare Ogunbowale, Olayinka Adetola, Na Li, Prairie View A&M University

Programming Many-Core Architectures (GPUs) Using CUDA — Conference Tutorial **85**
Eduardo Colmenares, Midwestern State University

Designing Learning Outcomes and Competencies using Bloom's for Computing — Conference Tutorial **86**
Markus Geissler, Koudjo Koumadi, Pam Schmelz, Christian Servin, Cara Tang, Cindy Tucker, Committee for Computing Education in Community Colleges

The Consortium for Computing Sciences in Colleges Board of Directors

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

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

Karina Assiter, Vice President/President-Elect (2024), KarinaAssiter@landmark.edu, Computer Science, Landmark College, Putney, VT 05346.

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

Brian Hare, Treasurer (2023), hareb@umkc.edu, School of Computing & Engineering, University of Missouri-Kansas City, Kansas City, MO 64110.

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

Judy Mullins, Central Plains Representative (2023), mullinsj@umkc.edu, University of Missouri-Kansas City, Kansas City, MO (retired).

Michael Flinn, Eastern Representative (2023), mflinn@frostburg.edu, Department of Computer Science & Information Technologies, Frostburg State University, Frostburg, MD 21532.

David R. Naugler, Midsouth Representative (2025), dnaugler@semo.edu, Brownsburg, IN 46112.

David Largent, Midwest Representative(2023), dllargent@bsu.edu, Department of Computer Science, Ball State University, Muncie, IN 47306.

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

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

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

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

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

Bryan Dixon, Southwestern Representative (2023), bcdixon@csuchico.edu, Computer Science Department, California State University Chico, Chico, CA 95929.

Serving the CCSC: These members are serving in positions as indicated:
Bin Peng, Associate Editor, bin.peng@park.edu, Department of Computer Science and Information Systems, Park University, Parkville, MO 64152.
Ed Lindoo, Associate Treasurer & UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.
George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood College, Frederick, MD 21701.
Megan Thomas, Membership System Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.
Karina Assiter, National Partners Chair, karinaassiter@landmark.edu, Landmark College, Putney, VT 05346.
Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

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

Platinum Level Partner

Google Cloud
GitHub
NSF – National Science Foundation

Gold Level Partner

zyBooks
Rephactor

Associate Level Partners

Mercury Learning and Information
Mercy College

Welcome to the 2023 CCSC South Central Conference

The 2023 South Central Steering Committee is very pleased to welcome everyone to our 34th annual conference hosted by Stephen F. Austin State University in Nacogdoches, Texas. Unlike the previous two years, our conference chair and host, Anne Marie Eubanks, has provided infrastructure and support for the in-person delivery of our conference this year.

For our 2023 conference, we have seven papers, three tutorials, and both student and faculty posters scheduled for the program. This year the Steering Committee chose 7 of 15 papers through a double-blind review process for a paper acceptance rate of 46%. Sixteen colleagues across the region and country served as professional reviewers and we recognize the expertise and guidance they all so thoughtfully contributed to the selection of our 2023 conference program.

The Steering Committee continues to seek colleagues to host the conference in the future and to join our community of computer science educators to enrich our curricula and provide innovative pedagogy for our students. We invite and encourage our fellow members of the South Central region to attend our steering committee business meeting on Friday, March 31, 2023 after the conference reception and banquet. Fellow educators and colleagues are encouraged to join in our efforts to involve more of our community in the planning and execution of the conference in the future.

We extend a very warm and delightful welcome to our presenters and attendees who continue to promote computer science education and camaraderie to our region. To all members of our 2023 Steering Committee, thank you again for your help in organizing the conference and your gracious efforts in delivering our conference during such challenging times.

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

Bingyang Wei
Texas Christian University
Regional Editor Co-Chair

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

2023 CCSC South Central Conference Steering Committee

Conference Chair
Anne Marie Eubanks ..... Stephen F. Austin State University, TX
Past Conference Chair
Shyam Karrah ..... University of Texas at Dallas, TX
Papers Chair
Bingyang Wei ..... Texas Christian University, TX
Mustafa Al-Lail ..... Texas A&M International University, TX
Reviewer Chair
Lasanthi Gamage ..... Webster University, MO
Julie Smith ..... University of North Texas, TX
Panels and Tutorials Chair
Jeffrey Zheng ..... Stephen F. Austin State University, TX
Posters Chair
Christain Servin ..... El Paso Community College, TX
Moderator Chair
Abena Primo ..... Huston-Tillotson University, TX
Publicity Chair
Eduardo Colmenares-Diaz ..... Midwestern State University, TX
Nifty Assignments Chair
Michael Kart ..... St. Edward's University, TX
At-Large Member
Tim McGuire ..... Texas A&M University, TX

Regional Board — 2023 CCSC South Central Region

National Board Representative
Tina Johnson ..... Midwestern State University, TX
Registrar
Anne Marie Eubanks ..... Stephen F. Austin State University, TX
Treasurer
Bilal Shebaro ..... St. Edward's University, TX
Regional Editor
Bingyang Wei ..... Texas Christian University, TX
Mustafa Al-Lail ..... Texas A&M International University, TX

Reviewers — 2023 CCSC South Central Conference

Laura Baker ..... St. Edward's University, Austin, TX
Eduardo Colmenares-Diaz ... Midwestern State University, Wichita Falls, TX
Andrea Edwards ..... Xavier University of Louisiana, New Orleans, LA
Lasanthi Gamage ..... Webster University, Webster Groves, MO
Thoshitha Gamage .....
. ..... Southern Illinois University Edwardsville, Edwardsville, IL
David Gurney ..... Southeastern Louisiana University, Hammond, LA
Tina Johnson ..... Midwestern State University, Wichita Falls, TX
Srinivasarao Krishnaprasad ... Jacksonville State University, Jacksonville, AL
Vipin Menon ..... McNeese State University, Lake Charles, LA
Jose Metrolho .....
. ..... Instituto Politécnico de Castelo Branco, Castelo Branco, Portugal
Muhammad Rahman ..... Clayton State University, Morrow, GA
Ken Rouse ..... Letourneau University, Longview, TX
Michael Scherger ..... Texas Christian University, Fort Worth, TX
Bilal Shebaro ..... St. Edward's University, Austin, TX
Bingyang Wei ..... Texas Christian University, Fort Worth, TX
Fei Zuo ..... University of Central Oklahoma, Edmond, OK

Keynote Speaker – Paul R. Gault

Paul R. Gault is a seasoned business executive and technology leader with over 25 years of experience in the software industry. He currently serves as the Senior Vice President of Customer Success and Technical Support at Provalus; whose mission is to elevate under-served communities by providing technology, business and support positions to untapped talent in the U.S. Their work provides Fortune1000 companies the dependable, quality and practical services they need... straight from the heart of America.

Prior to joining Provalus, Paul held several executive positions at various software companies, including Advisory Specialist Leader for Deloitte, Director of Security Solutions for InteliSource, Senior Network Administrator for General Dynamic Information Technology, and over 20 years in the United States Airforce as Superintendent, Standards and Self-Assessments/Quality Assurance and Network Team Lead, Integrated Network Operations and Security Center. In these roles, he was responsible for building and scaling global support organizations that provided mission-critical services to enterprise customers.

Paul is a results-driven leader with a passion for delivering exceptional customer experiences. He is known for his ability to build high-performance teams, foster collaboration across departments, and drive innovation in support delivery. He is also a strong believer in using data and analytics to measure customer success and optimize support operations. He has a strong passion for disaster and humanitarian relief, economic empowerment, politics, science and technology, and education.

Supporting Low-Income, Talented Undergraduate Students in Engineering and Computing Sciences with Scholarships and Mentoring*

Dulal C. Kar, Scott A. King, and Dugan Um
Texas A&M University-Corpus Christi
Corpus Christi, TX 78412
{dulal.kar, scott.king, dugan.um}@tamucc.edu

Abstract

The NSF S-STEM program supports low-income, talented students seeking an education and career in STEM fields. This works presents the results of an NSF S-STEM grant awarded to Texas A&M University-Corpus Christi, a Hispanic Serving Institution, that recruited and supported 39 talented and financially needy undergraduate students including 23 students from underrepresented groups. Each student was mentored, guided, and supported with curricular and co-curricular activities for engineering and computer science majors and relieved from financial burden of paying tuition and other expenses with an amount of $7,000 per year. There were 17 community college transfer students and 22 high school seniors recruited altogether. Despite the support of the scholarship, seven students could not continue in the program since they could not maintain the GPA of 3.0 required to stay in the program. All seven dropouts were from the group of 22 recruited high school seniors, and none were from the group of the 17 community college transfer students. There were 11 students in the Hispanic students group. It was found that the overall GPA of the Hispanic students group went down near the end of their graduation while the overall GPA of the rest of the students improved slightly. In various support services and activities, midterm mentoring was found to be most helpful for students at-risk to

continue in the scholarship program and the participation in undergraduate research was found to be beneficial for some students in securing employment in leading industry. The findings and results of this S-STEM scholarship project are important to prospective S-STEM PIs (Principal Investigators) who plan to apply for NSF S-STEM grants and recruit Hispanics, women, and/or community college transfer participants.

1 Introduction

The NSF S-STEM scholarship program supports talented, needy undergraduate students with scholarships in S-STEM disciplines. The S-STEM program has helped many needy students around the nation to complete their education in a STEM discipline and then eventually enter a STEM career to contribute to advancement and applications of science, technology, engineering, and mathematics [9, 21, 11, 4, 1, 7, 14, 18]. Texas A&M University-Corpus Christi (TAMUCC) as a Hispanic Serving Institution received an S-STEM grant from NSF to offer scholarships to talented and disadvantaged undergraduate students majoring in engineering or computer science. The objectives of the S-STEM project were to recruit talented, needy undergraduates for majors in engineering and computer science disciplines, particularly as many as possible from underrepresented groups such as Hispanics and women and support them with mentoring, co-curricular, and professional development activities. Specifically, we targeted to recruit students not only from high schools but also from community colleges since many needy students attend a community college to further their education.

Community College Transfer Students: The ever-increasing cost of education in four-year degree institutions has made college education unaffordable for many low-income high-school graduates. Many of them have found low-cost community colleges to further their education beyond high school. The American Association of Community Colleges (AACC) reports that the annual average cost of tuition and fees for students in community colleges in the United States is about $3,300. In contrast, the annual average cost of tuition and fees at state funded institutions is approximately $9,000. Some have found community colleges as affordable pathways to higher education first by completing a two-year associate program at a community college with less cost in tuition and fees and then transferring to a four-year institution that gives credits earned in a community college [8, 16, 2, 15, 17]. However, upon completion of a community college degree, the additional increased cost of attending a four-year degree institution for the remaining two or three years of education can discourage a vast majority of economically disadvantaged students from pursuing a four year degree. Those who transfer from a community college to a four-year institution face severe financial hardship due to increased tuition

and fees and often have to work long hours in jobs to survive financially. The lack of time for study can cause poor academic performance for many of these individuals which in turn causes many to drop out from the pursuit of a four-year degree. Another consequence is delayed graduation as few courses are taken or courses with poor performance are repeated. The major goal of our NSF sponsored S-STEM scholarship program was to recruit and support community college transfer students financially in their pursuit of higher education by alleviating their need to work in low-paying, long-hour jobs and facilitating them with necessary time for study to be successful in their coursework.

Diversity and Recruitment of Hispanics: TAMUCC is a Hispanic Serving Institution with an enrollment of over 10,000 students, of which about 48% are Hispanic. However, Hispanic student enrollment percentages do not match the region's 60% Hispanic population. Nationally, Hispanics remain considerably underrepresented in engineering and computer science careers [13]. In 2017, ASEE reports that among the engineering BS degrees awarded, only about 11.1% were awarded to Hispanic students. As noted earlier, partly due to financial hardship, many Americans, and most Hispanics in Texas, choose to attend community college for their education beyond high school. However, a National Academies report found that students entering community colleges do not realize that they can obtain a four-year STEM degree such as in computer science or engineering by transferring to a four-year institution [12]. Earlier studies have indicated only about 20% of college-qualified low-income students transfer and attain a bachelor's degree. Hispanic students are the most likely to leave college without attaining a degree in large part due to financial burdens and cultural dislocation. It is to be mentioned that Hispanics are less likely to accept loans and take on that debt, and instead, they choose to work [20]. Similarly, another well-known underrepresented group in our recruitment target for the S-STEM program was women. It is well-known that enrollment of females in engineering and computer science is very low nationally. An important focus of our S-STEM scholarship project was to recruit as many talented students as possible from underrepresented groups, including Hispanic students and women.

Providing financial support to these students does not automatically guarantee their success in achieving a bachelor's degree from a four-year institution. There is a significant disparity between the number of underrepresented students who begin in the engineering and computer science disciplines and the number who complete their degrees eventually. Many Hispanic or Latino students are first generation college students who do not receive much motivational support from their family, who may not understand the struggles these students are going through [5]. These students experience a multitude of difficulties to adapt and integrate with the mainstream student body. Support services are

needed to help them succeed in their pursuance of a four-year degree in STEM disciplines [3, 6, 22, 19, 10]. Therefore, in addition to the financial support, our S-STEM scholarship project provided the academic, professional development, and emotional supports necessary to help these academically talented students succeed.

In this paper, we present our experience based on our NSF S-STEM grant that awarded $7,000 per year as a scholarship to each participant, which covered tuition and fees to alleviate the financial burden. It is to be mentioned that the current average cost of in-state tuition and fees at our institution for an undergraduate student is $9,292. This S-STEM project recruited altogether 39 students, in which 17 students were community college transfer students and the remaining 22 students started their college education at the institution directly after graduation from high school. Among the 39 recruited students, 11 students were Hispanic, and 13 students were female. The diversity of students in the S-STEM project provided us the opportunity to share our valuable experience with the S-STEM community, particularly in engineering and computing. The grant did not have any research requirement other than supporting participants with scholarships, supervision, and engaging them in voluntary learning and professional development activities.

2 Recruitment Efforts

To support recruitment of participants as well as to disseminate information about the S-STEM scholarship opportunity at our institution, a website with essential information needed to know by an applicant was maintained. The website provided information on the benefits of the program, eligibility requirements to apply, documents to be submitted with the application including: transcripts, an essay, and references. An application form was developed to receive information such as on the applicant's academic background, ethnicity, and work experience. A reference form was developed to receive feedback from the references on the applicant's skills, motivation, and even economic situation, if known. A brochure on the S-STEM opportunity was developed with requisite information such as the eligibility requirements, the scholarship amount, professional development opportunities, and support services.

An outreach and recruitment institutional representative from TAMUCC visited local and regional high schools and community colleges giving talks, distributing flyers, and meeting high school counselors and community college officials. TAMUCC has yearly designated days (official preview days) to conduct recruitment events on the campus designed especially for prospective undergraduate students and friends and family members to learn about the student life at the campus. Flyers on the S-STEM scholarship opportunity

were handed out to the attendees during those events. Also, during incoming freshmen mentoring sessions and orientation sessions flyers were distributed to the incoming freshmen in engineering and computer science programs.

The institutional outreach and recruitment representative attended career fairs for high school students in Houston, Dallas, and Rio Grande Valley areas and distributed flyers on the S-STEM scholarship opportunity. In regional "Counselor Updates" organized in Dallas, Houston, and Rio Grande Valley areas, admission counselors of our institution provided high school and community college counselors information about the S-STEM scholarship opportunity. An application package would include the application form, a resume, transcripts, an essay, and three completed reference forms. A rubric with evaluations points on GPA, quality of the essay, standardized test score (SAT or ACT sore), work experience, and reference information was developed to evaluate applications by a selection committee. For each applicant, an overall total score was calculated by assigning a weight to each score summing all weighted scores. The rank of each applicant was determined based on the overall score calculated. The recruitment decision on the number of participants for a program was made in proportion to the number of students enrolled in each program, either in engineering or computer science. Besides, ranking, diversity was also considered while selecting participants for the S-STEM project.

The original project duration was five years, which was extended to seven years with approval from NSF. Over the period of the project, there were 39 students altogether recruited in the program including: White – 25, Hispanics – 11, African American – 1, Asian – 1, and Unreported Ethnicity – 1. Each of the participants was either a U.S. citizen, permanent resident, national, or refugee, as required by NSF, and each of them had a GPA of at least 3.0 at the time of recruitment. By gender, there were 13 women among the participants, exactly one-third of the total 39 participants. As stated above, according to our recruitment plan, we recruited 17 community college transfer students. Each student was supported with a scholarship of $7,000 per year for up to four years for a regular undergraduate. The support period was shorter for community college transfer students.

3 Support Services and Activities

Earlier studies show that mentoring contributes to the retention and academic success of students [4, 6, 22, 15]. The S-STEM participants received academic and career support using three major mentoring strategies: 1) Faculty Mentoring, 2) Academic Counseling, and 3) Career Counselling. A team of faculty and staff met regularly with the participants to promote academic success, retention, and progress to graduation and employment. It is to be noted that

in compliance with NSF S-STEM scholarship program requirements, participation in any curricular, co-curricular, or professional activities for a student was voluntary.

At the beginning of each semester, we would organize a mentoring session or retreat. Attendance to the retreat was mandatory for all students. To handle scholarship offers to participants as well as to oblige them to fulfill the requirements of the scholarship award, a contract document on the scholarship award was used. The contract document includes some terms and requirements such as being a full-time student in computer science or engineering, maintaining satisfactory academic progress (GPA $>$ 2.9), completing FAFSA, attending mandatory meetings, and attending mentoring sessions and seminars. During each fall semester retreat, each participant would sign the contract of the scholarship containing a clause in the contract regarding the GPA requirement to continue in the program and to participate voluntarily in many activities and take advantage of available resources in the campus such as for tutoring, academic counseling, and career counseling needed to succeed in the pursuit of the BS degree. In each retreat, the participants would receive useful information from academic counselors, career counselors, financial aids office administrators, and peer participants.

During the retreat, the participants were reminded about the mission of the NSF S-STEM program and its significance at the national level in advancement of science, engineering, and mathematics. In this context, the purpose, goals, objectives, and expectations of the S-STEM scholarship project at TAMUCC were made known to the participants. Requirements in terms of academic performance and participation in activities were informed.

Focusing on Education: Career prospects in engineering and computer science as well as strategies to succeed in college education were discussed in retreats. It was observed that many scholarship recipients continued working and falling in danger of loosing the scholarship. They were advised to concentrate on education and graduate on time, rather than working part-time to supplement income for living expenses and graduate later. Job prospects for students graduating in engineering and computer science were highlighted to raise the level of their motivation. They were reminded that the scholarship support paying their tuition and fees should relieve them from working long hours in jobs and concentrate on study and they should not seek any jobs to support themselves while seeking college education. They were informed about availability of various support services within the campus such as the programming assistance lab, the writing center, and the tutoring center, and encouraged to take advantage of these services to advance their education.

Balancing Course Load: Advising students with a balance of intensive, non-intensive, hands-on, technical, and non-technical courses is critical to stu-

dent success and hence to retention. The scholars were instructed how to choose courses based on technical, analytical, and hands-on contents and accordingly, enroll in courses each semester to achieve academic goals each time successfully. They were advised to form study groups to succeed in difficult courses, study objectively from presentation materials used in classes with questions in mind, and develop problem solving skills in engineering, computer science, and mathematics by working on example problems step by step. Helpful tips were provided how to do well in college, how to manage time for various activities, and how to make use of support facilities available within the college.

Memberships in Professional Societies: Joining professional societies is very helpful for professional development. The participants were encouraged to join professional societies and informed about the availability of fund to pay for their memberships of professional societies such as IEEE, ACM, and ASME.

Undergraduate Research Opportunities: Importance of undergraduate research was stressed, particularly for a better job and career prospect. All participants were informed about the research opportunities available on campus and were encouraged to contact the engineering or computer science faculty members to join their research teams. TAMUCC supports many STEM undergraduates each year for research and travel to conferences through an NSF sponsored LSAMP (Louis Stokes Alliances for Minority Participation) grant. The S-STEM participants were informed to apply for the research opportunity. The participants were introduced with NSF sponsored REU programs offered at different institutions around the nation. We provided references to the participants who applied for this opportunity. In addition, the participants were encouraged to apply to the NSF sponsored computer science REU program offered at our institution. We demonstrated to the S-STEM scholarship recipients how to access NSF websites for engineering and computer science summer REU programs, explore NSF-funded REU programs in various institutions including the ones in the state of Texas, and find out specific information and benefits. Benefits of these programs including research experience, professional growth opportunities, stipends, and allowances for travel, food, and housing were stressed. The S-STEM participants were shown how to apply online for various REU opportunities. Several participants joined the REU program offered at our institution and some other institutions around the nation.

Career Services: One of the career counselors of our institution would give a presentation at each retreat on various resources and services available at the institution. Participants were informed on how they can use online tools and resources available for career planning (CHOICE360, BLS.GOV, and OnetOnline.org), preparing professional documents, job hunting, preparing for graduate/professional school, and networking with professionals. Internship

opportunities and benefits were discussed. The S-STEM scholars were informed about professional etiquette to be followed during job interviews and asked to participate in mock interviews for practice. They were invited to attend career fairs organized at our institution throughout the year. During retreats, other presentations by University Services Senior Personnel include presentations by a financial aid advisor of the financial aid office, presentations by academic advisers of the College of Science and Engineering, and presentations by the directors of student support services, tutoring, learning, and writing centers.

Presentations by Peer Students: Several scholars who participated in undergraduate research programs including NSF-sponsored REU programs and presented papers or posters in conferences also gave presentations during retreats. Similarly, some of the scholars who did summer internships in regional industry also gave presentations to the participants during mentoring sessions or retreats. Motivated by these presentations, many of them applied for REU and internship opportunities.

Beyond retreats, the following support services were provided to the S-STEM participants on a regular basis or as needed.

Faculty-Student Mentoring: Mentor training is provided by our institution's Office of Student Engagement and Success. Faculty mentors are informed at the beginning of each academic year how to access online and other resources available at the institution to successfully mentor students. Information on student support services and mentoring models is the primary focus of the training. It is understood that the mentoring needs of a student vary on ethnicity and socio-economic background as well as the academic level of the student. Faculty members are informed and trained accordingly with information and approaches needed to successfully mentor students of diverse ethnic background and academic level. A separate group of faculty members is responsible to mentor freshman-level students, particularly in the computer science program.

General Counseling: The S-STEM participants had access to individual counseling, personal skills training focused on helping students improve goal setting, memory and study skills, workshops on topics such as wellness, healthy relationships, and assertiveness, plus many other topics. A number of services and computer software programs are available for free via the University's Transfer Center. S-STEM participants were recommended to take advantage of such services that included career selection and skill enhancement in note-taking, test strategies, time management, stress management, financial management, and public speaking.

Professional Interactions: As mentioned above, the S-STEM participants are informed about the benefits and the funding available to become members of one group of their choice, such as ACM Computer Science Club,

American Society of Mechanical Engineers Student Chapter, and Association for Women in Science. Time to time, these student organizations would bring in their meetings industry professionals for seminars and short talks thus providing them opportunities to learn about career prospects in industry.

Midterm Mentoring: Midterm mentoring was found to be the most beneficial support service for the students who were at risk of losing the scholarship due to low GPA. After the initial mentoring session or retreat each semester, there was a mid-term mentoring/advising meeting with some selected at-risk students. In this effort, only the students who were at-risk with low mid-term grades were contacted. The project leadership would meet each student individually and check midterm grades reported by faculty in all courses. Students having trouble in some courses were advised to meet course instructors and attend tutorial services provided by the university as well as provided with helpful tips such as to form study groups and join online forums. Occasionally, a course instructor was also contacted to setup a meeting with the affected student to receive helpful guidance to improve performance.

The first step in midterm mentoring was to find out the reason behind poor performance in a course. Often, students would reveal their struggle in their personal lives, and they would need to be directed to receive general counseling. For directly course related situations, depending on the situation, they were advised to meet with the course instructor to find out how to improve in the course, form a study group to improve understanding of course material, seek help from teaching assistants in programming assignments, visit the tutoring center on math or physics assignments, and make appointments with the writing center to improve writing. In some cases, it was found that a student was working long hours which was the main cause of poor performance in multiple courses, and it was necessary to advise the student to attain some balance between work and study. A powerful argument was to appeal to them that if they could not spend enough time on their studies, they were putting their scholarship at risk and, also that they should look at their scholarships as if they were being "paid" to study, and they, therefore, needed to fulfill that requirement. It was found that working long hours was one of the primary reasons for poor performance for many participants. After mind-term mentoring, it was found that some students would reduce their work hours to make more time to improve their grades. In some cases, it was necessary to help students by developing a well-balanced schedule of work and study as well as allocating ample time in the schedule to study for improving grades in difficult courses.

The NSF S-STEM program collects data on each S-STEM project on activities. Table 1 summarizes data collected in our project on all 39 students who were recruited and supported through the S-STEM project. It is to be noted that attending a retreat or mentoring session was mandatory for all par-

ticipants and hence the corresponding participation rate was 100%, as shown in the table. The "Other" activities include many miscellaneous activities such as joining a club, playing sports, providing tutoring services, helping in church services, and so on.

Table 1: Participation in Activities

Item	Academic Year						Yearly
	2015-2016	2016-2017	2017-2018	2018-2019	2019-2020	2020-2021	Rate
No. of Participants	7	14	13	30	24	16	-
Part-Time Employment	3	5	6	17	12	9	50%
Academic Support Services	4	5	8	19	15	10	59%
Career Counseling	2	2	7	12	9	8	38%
Community Building	0	5	7	7	5	2	25%
Field Trips	0	2	3	7	6	1	18%
Internships	1	0	1	9	3	1	14%
Meetings/Conferences	2	7	4	17	15	9	52%
Mentoring	7	14	13	30	24	16	100%
Recruitment	0	0	2	5	3	1	11%
Research	0	1	1	7	4	3	15%
Seminars	4	4	3	27	7	3	46%
Other	4	5	8	17	15	5	52%

4 Results and Discussion

As mentioned above, of the 39 students in the program, there were 23 students (59% of the total number of participants) altogether from underrepresented groups including women. In terms of gender, there were 13 women among the participants. Ethnic minorities include 11 Hispanics, one African American, one Asian, and one student of undeclared ethnicity. In all, 29 students completed the S-STEM program successfully, including all 17 community college transfer students. Among the ten students who could not or did not continue in the program, seven students could not maintain the required minimum GPA of 3.0, one student switched to a different STEM major, one student transferred to a different institution and continued in a STEM major, and one student dropped out from our institution after adversely affected by a natural disaster (hurricane). Overall, the attrition rate is 18% (7 out of 39) due to low GPA. Over the period of the grant, 35 students have graduated with a BS degree either in engineering or computer science or in a STEM discipline, i.e., the graduate rate is about 90%, thereby increasing the pool and diversity of the nation's workforce in these critical STEM fields. Table 2 shows overall average GPAs of different groups of students. We observe improvement in the academic performance of the community college transfer students as indicated in their overall final average GPA. In contrast, the performance of high school seniors recruited in the program shows a decline in overall final average GPA by 2%.

In terms of gender, almost no change in academic performance is observed. A significant decline of 3% in overall average GPA in academic performance in the Hispanic ethnic group is found.

Table 2: Academic Performance by Ethnicity, Gender, and Initial Level of Education

		Initial Avg. GPA	Final Avg. GPA	Interpretation
Ethnicity	Hispanic	3.74	3.63	-3% (decrease in GPA)
	White	3.53	3.54	No significant change
Gender	Female	3.65	3.66	No significant change
	Male	3.55	3.53	No significant change
Group	High School Senior	3.69	3.61	-2% (decrease in GPA)
	Community College Transfer	3.54	3.60	2% (increase in GPA) in GPA

In the following, we summarize our findings and make suggestions that can be helpful for all participants, especially low-income, talented participants of Hispanic descent:

1. **Scholarship Amount:** An amount of $7,000 per year was not enough as a scholarship for most students as they had to work long hours to support themselves. Some students were found to hold jobs for 30 hours per week. Increasing scholarship amount will help these students to devote more time to study, continue in the program, and graduate on time. The maximum scholarship amount of $10,000 is allowed by the NSF S-STEM program. It is recommended that the maximum amount should be given as a scholarship to each participant.

2. **Midterm Mentoring:** Midterm mentoring was very effective in retaining the students in the S-STEM project, particularly the students at risk. Most students reported with midterm grades of C or D in courses were eventually able to attain better grades or to maintain the overall GPA of at least 3.0. More interventions such as midterm mentoring were found to be beneficial to improve academic performance of the participating students. For example, in Fall 2018, among 29 students, 17 students had a grade of C or D in one or more courses. These students were given guidance how to improve their situation in those courses. All these students did well and were able continue their scholarships in spring 2019 with an overall GPA greater than 2.9. Similarly, in spring 2019, among 25 students, eight students had a grade of C or D in one or more courses. Except one student, all other seven students at-risk were able to maintain a GPA above 2.9.

3. **Research Participation:** Several students involved in undergraduate research, including in NSF REU programs, were found to receive employment in leading industry or seek graduate education. In surveys, many participants expressed positively on the impact of research experience in their academic pursuit, employment success, or career path.

4. **Learning Community:** The dropout rate of high school senior recruits from the S-STEM program was found to be too high. One possible solution is to enlist them in learning communities/cohorts and guiding them with advice more frequently [19, 10].

5. **Professional Development:** In compliance with NSF's requirements, participations in co-curricular activities were voluntary for students. Accordingly, the students were advised to attend seminars, symposiums, and conferences as well as to join internships and undergraduate research programs. However, a focus group study reveals that many participants wanted to have more engaging professional development activities through workshops such as on Presentation Skills, Technical Writing, Resume Preparation, and Ethics. Due to lack of funding in the S-STEM grant to support such activities at the time, it was not possible for us to organize sessions or workshops.

Focus Group Study: A focus group study was conducted by an external evaluator. The focus group participants were predominantly sophomores (5) and juniors (4). One female senior participated. There were six male informants and four females. The focus group participants agreed that benefits of the S-STEM program fell in two categories, financial and motivational. They stated that without the scholarships they would have needed to work to complete school and felt that would have impacted their ability to be successful. The focus group participants said that the S-STEM scholarships gave them a sense of accomplishment, indicated to them that they were capable and that they could do more than they might otherwise have thought, and that the awards gave them motivation to attend classes, to stay "on top of things; classes and grades, and that the GPA requirement motivated them to invest time and effort in meeting that standard. The most valuable aspect(s) of the S-STEM project, as stated by the focus group participants, are: 1) the ability to continue as an undergraduate student in engineering or computer science major, 2) the ability to work toward a degree without incurring debt or, at least, with a limited amount of debt, 3) the meetings held each semester at which participants were reminded of project commitments and opportunities that were before them, 4) the opportunities presented to meet faculty members, and 5) being encouraged by project personnel.

5 Conclusion

This paper presents the results and experiences of an NSF S-STEM grant that supported 39 undergraduate students in which there were 17 community college transfer students, and the rest were recruited high school seniors. Among the participants, there were 11 Hispanic students and 13 female students. Seven students could not continue in the program as they could not maintain a GPA of at least 3.0. Altogether, 35 students have graduated successfully including three students who were dropped out from the S-STEM program due to low GPA. The most of the dropout students were found to be busy working long hours to support themselves financially. Increasing the scholarship amount could help these students spend more time in education than in work. The overall performance of the community college transfer students was better than the students recruited from high school. All the attritions were from the group of the recruits from high school. Some better support strategies such as enrolling students in a learning community or preparatory summer courses in cohorts can help. Midterm mentoring was found to be helpful for students at risk to continue in the program. Several students who participated in undergraduate research were successful in obtaining employment in leading companies in the nation.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant DUE-1458096. 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.

References

[1] Joel C. Adams and Randall J. Pruim. Computing for stem majors: Enhancing non-cs majors' computing skills. In *Proc. SIGCSE*, pages 457–462, Raleigh, NC, 2012.

[2] Oscar S. Cerna, Patricia A. Perez, and Victor Saenz. Examining the precollege attributes and values of latina/o college graduates. In *Higher Education Research Institute: Research Report*, Los Angeles, CA, 2007.

[3] Martin M. Chemers, E.L. Zurbriggen, M. Syed, B.K. Goza, and S. Bearman. The role of efficacy and identity in science career commitment among underrepresented minority students. *Journal of Social Issues*, 67(7):469–492, 2011.

[4] Teresa J. Cutright and Edward Evans. Year-long peer mentoring activity to enhance the retention of freshmen stem students in a nsf scholarship program. *Mentoring & Tutoring: Partnership in Learning*, 24(3):201–212, 2016.

[5] Jennifer M. Dennis, Jean S. Phinney, and Lisa I. Chuateco. The role of motivation, parental support, and peer support in the academic success of ethnic minority first-generation college students. *Journal of College Student Development*, 46(3):223–236, 5 2005.

[6] Susan Gershenfeld. A review of undergraduate mentoring programs. *Review of Educational Research*, 84(3):365–391, 2014.

[7] Heather B. Gonzalez. An analysis of stem education funding at the nsf: Trends and policy discussion. Technical report, Washington D.C., 2012.

[8] Linda S. Hagedorn and Astrid V. Purnamasari. A realistic look at stem and the role of community colleges. *Community College Review*, 40(2):145–164, 2012.

[9] Sylvia M. James and Susan R. Singer. From the nsf: The national science foundations investments in broadening participation in science, technology, engineering, and mathematics education through research and capacity building. *CBE—Life Sciences Education*, 15(3):1–8, 2016.

[10] Maria Kalevitch, Caleb Maurer, Paul Badger, Glenn Holdan, Joseph Iannelli, Ahmet Sirinterlikci, George Semich, and James Bernauer. Building a community of scholars: One university's story of students engaged in learning science, mathematics, and engineering through a nsf s-stem grant. *Journal of STEM Education*, 16(2):40–45, 2016.

[11] Junalyn Navarra-Madsen, Rodney A. Bales, and DiAnna L. Hynds. Role of scholarships in improving success rates of undergraduate science, technology, engineering and mathematics (stem) majors. *Procedia - Social and Behavioral Sciences*, 8:458–468, 2010.

[12] National Academy of Sciences, National Academy of Engineering, and Institute of Medicine. *Rising Above the Gathering Storm, Revisited: Rapidly Approaching Category 5.* National Academies Press, Washington, DC, 2010.

[13] John Roy. Engineering by the numbers, 7 2019.

[14] Domenico Russomanno, Richard Best, Sarah Ivey, John R Haddock, Donald Franceschetti, and Rebecca J Hairston. Memphistep: A stem expansion program at the university of memphis. *Journal of STEM Education*, 11(1), 2010.

[15] Carol Slater, William Edmister, Brenda Watford, and Joel Kampe. Lessons learned: Implementing a large-scale peer mentoring program. *Proceedings of the ASEE Annual Conference and Exposition*, 2006.

[16] Tom Springer, Callam K. Girolami, and W.K. Kellogg Foundation. *ENLACE Connection: What Makes a Difference in the Education of Latino U.S. Students–Learning from the Experience of 13 ENLACE Partnerships*. W. K. Kellogg Foundation, Battle Creek, MI, 2007.

[17] Michael Summers and Freeman A. Hrabowski. Preparing minority scientists and engineers. *Science*, 311(5769):1870–1871, 2006.

[18] Vincent Tinto. Taking retention seriously: Rethinking the first year of college. *NACADA Journal*, 19(2):5–9, 1999.

[19] Vincent Tinto. What have we learned about the impact of learning communities on students? *Assessment Update*, 12(2):1–4, 2002.

[20] Francisco Vara-Orta. Most latino students spurn college loans. *The Los Angeles Times*, 2007.

[21] Zakiya S. Wilson, Sitharama S. Iyengar, Su-Seng Pang, Isiah M. Warner, and Candace A. Luces. Increasing access for economically disadvantaged students: The nsf/csem & s-stem programs at louisiana state university. *Journal of Science Education and Technology*, 21(5):581–587, 2012.

[22] Daniel Yomtov, Scott W Plunkett, Rafael Efrat, and Ana Gabriela Marin. Can peer mentors improve first-year experiences of university students? *Journal of College Student Retention: Research, Theory & Practice*, 19(1):24–44, 2017.

Iterative Efforts for Improving Learning Experience in Software Engineering*

Pradip Peter Dey, Mohammad Amin and Bhaskar Raj Sinha
School of Technology and Engineering
National University
9388 Lightwave Ave., San Diego, CA 92123
{pdey, mamin, bsinha}@nu.edu

Abstract

In a project-based learning environment, students and teachers jointly made iterative efforts for improving learning experience in software engineering through all major tasks including requirements analysis, design, implementation, and testing. The iterative efforts were implemented in a prototype-based evolutionary process by performing reviews jointly by students and teachers after each major task, and assessing student performance based on their participation in task-related activities. End of course evaluation data, collected in a standard anonymous process, indicated improvements in student learning experience and teaching effectiveness attributable to the iterative efforts. The major advantages of the iterative efforts were engaging students in the review process, and eliminating or reducing plagiarism-based academic dishonesty by emphasizing participation-based grading. One of the major challenges for teachers was making extra efforts for participation-based grading, rather than using automated grading of multiple choice exams and quizzes. In addition, extra efforts were needed to complete three iterations for sizable software engineering projects in a timely manner in order get benefits of iterative efforts in project-based learning environments. There are opportunities for future research in this area for creating a set of revealing software engineering projects of appropriate sizes and explaining their potential benefits in teaching learning environments.

1 Introduction

In a project-based learning environment [6], teachers and students can work together cooperatively performing software requirements engineering, design, implementation and testing, and their reviews iteratively in order to promote desirable learning experience [16] and fair grading. Key development skills are in demand because developers are facing increasingly more complex problems in their work environment. Design skills are more important than other skills, because "The ability to recognize complexity is a crucial design skill" [11] (page 5). When a designer recognizes that a system is too complicated, the designer can use that ability to guide their design philosophy towards dealing with the complexity [11]. Designers should be able to explain how the software elements will work together collaboratively in operational scenarios to provide appropriate services to users with excellent user experience.

Great designers such as Steve Jobs and Jony Ive (of Apple, Inc.) have not described their innovative design approaches for teaching-learning purposes [8]. Jason Hong has studied the practices in Apple, and asked the following question "Why is great design so hard?" [7]. In this paper, we critically examine some teaching strategies for making iterative efforts for improving learning experience in design, and suggest that one of the strategies has great potential in project-based learning [6]. For explaining the potential, we emphasize graphical user interface (GUI) design. Evidence from end of course evaluation data, collected in a standard anonymous process, is presented that indicates improvements in students learning experience and teaching effectiveness attributable to iterative efforts.

2 Iterative Efforts

Iterative efforts for software engineering can be made with well-known process models such as evolutionary, spiral, agile, or prototype-based development models [3, 5, 9, 11, 12, 13, 14, 15, 17]. Iterative efforts using evolutionary prototypes are often recommended in object oriented software development [9, 14]. One can follow Pressman and Maxim's textbook which suggests that "Software design is an iterative process . . ." [13] (page 228). Designers can make excellent progress through successive refinement in an iterative process. Most process models allow design reviewers and designers to ask questions such as, "Is this the best possible design for this product? Or, are there opportunities for improvement?" Iterative efforts with these types of questions promote innovations as demonstrated by Steve Jobs and Jony Ive [8]. Effective GUI designers repeatedly examine their designs in order to make improvements where many alternatives are carefully considered. GUI design and development can-

not be done in a hurry considering only some obvious alternatives. Experiments with evolutionary prototypes and their careful reviews may focus on providing excellent user experience [9, 13].

3 Reviews in Software Engineering

All major tasks in software engineering should be reviewed for preventing errors or defects. Software requirements engineering, design, implementation, testing and their reviews may be introduced to students using examples and textbooks [9, 13]. A good strategy for initial design is to decompose the problem into several sub-problems or relatively independent conceptual elements [11]. However, this strategy requires some understanding of interactions among the conceptual elements. Use cases and their relationship from requirements engineering may help at this stage [13, 14]. Following the Unified Modelling Language (UML), one can also draw use case diagrams [13, 14]. Preliminary reviews can start with the use case diagrams. For better reviews, other UML diagrams including component diagram, class diagram, sequence diagram, state machine diagram, and activity diagram may be considered [13, 14]. A use case diagram presents major use cases with ovals in a box or rectangle displaying actors outside the box to indicate that the actors are external users of the current system. Each use case represents a case of use that can be further illustrated in other UML diagrams such as activity diagrams, sequence diagrams [13, 14]. Consider a sample use case diagram shown in Figure 1; it was drawn in the UML 2.0 notation with a minor extension for a sample software project, which was initially described as follows:

"Develop a software system for computing areas of three types of play-place units: Rectangular, Circular and Triangular. A contractor in Los Angeles builds play-places (with materials such as wood, iron, pads, plastics etc.) at customer site using play place units of different dimensions. The charges are in dollars based on the area of each unit in square feet, plus the number of units. The software system is needed for computing the cost which is based on area. The cost is five dollars per square foot. Assume that users always use feet for entering the dimensions of the units. A Graphical User Interface (GUI) is required for user interactions. Additional typical assumptions can be made about this project."

In addition to the above sample problem, Albrecht's function point method [1] was used for explaining iterative efforts (a prototype for function point estimation is available at http://www.asethome.org/project/functionpoint.html). Iterative efforts for major tasks and their reviews were explained with two textbooks [9, 13]. The interfaces shown with the dotted rounded rectangles in Figure 1 were explained with special care, because these were absent in

the standard UML use case diagrams [13, 14]. These dotted interfaces were called general interfaces in order to distinguish them from specialized UML interfaces such as provided interfaces and required interfaces [14]. In order to conveniently refer to the general interfaces, they were sequentially numbered. A general interface shown within the system box must be developed as a part of the current software system; otherwise, it should be depicted outside the system boundary (such an interface is usually available in the development environment). If an interface was a GUI, then it was marked with the term "GUI" utilizing UML stereotypes [14].

It was reasonable to be flexible about the notations of the diagrams, because we needed to emphasize design skills, and not necessarily the diagram notations. Practitioners in agile development techniques value "Working software over comprehensive documentation" [4]. Following the "Manifesto for Agile Software Development", we emphasized working software that may satisfy end-users in operational environments [4]. In our project-based environment, the primary measure of progress in software development was working software, not extensive documentation with precise diagrams in standard notations. We considered two main alternative notations for the general interface diagrams: (1) screen shots from a prototype, and (2) abstract graphical representation of major interface elements. We show the former notation in the general interface diagram given in Figure 2 for the general interface 1 of Figure 1. The GUI in Figure 2 is from our third iteration.

4 Guidelines for Design

As designers and design-reviewers go through the iterative process they may like to follow some guidelines. Most of the guidelines suggested below came from Pressman and Maxim [13]; others were inspired by Hong [7], and Kung [9].

1. Examine promising alternatives from the widest range of possible alternatives in order to provide the best user experience through integration of various features including artistic, mathematical and intuitive aspects.

2. Utilize Object Oriented Design concepts throughout the development process.

3. Consider separation of concerns in order to deal with complexity of the system and interactions among system elements.

4. Consider design principles as well as human-computer interaction (HCI) data and user experience for innovative user interface solutions.

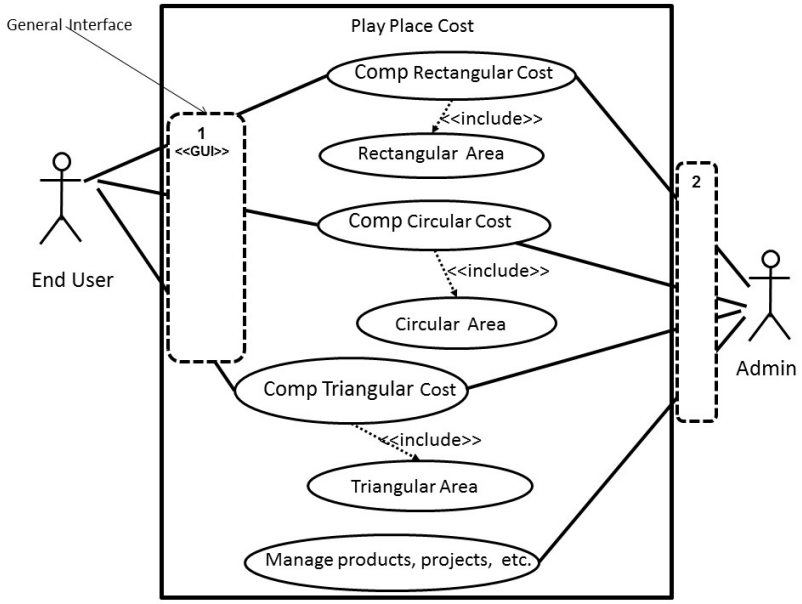


Figure 1: Augmented use case diagram with general interfaces.

Figure 2: A prototypical general interface diagram (from third iteration)

5. Include only those action features which are intuitively learnable; transform others to an automated category.
6. Maximize cohesion and minimize coupling among components.
7. Include error prevention and simple error handling.
8. Present design models at multiple levels of abstraction.
9. Make iterative efforts through design-review-redesign as many times as possible using the guidelines 1 through 8.

With reference to guideline 1, innovative designers often consider many unusual alternatives in addition to obvious ones. Quick design in a hurry may lead to consideration of only a few obvious alternatives missing innovative solutions [7, 8]. Steve Jobs and Jony Ive came up with brilliant user interface solutions that were missed by other practitioners in the same domain [7, 8]. Object-oriented design of guideline 2 is emphasized in several popular books [9, 13]. Object-oriented design elements such as fields, windows, buttons, allow rapid prototyping [9, 13]. Guideline 3 is essential for solving complicated problems [11, 13]. Guideline 4 is based on a reasonable integration of HCI factors [15], user experience, and other advanced design strategies [13, 14, 15]. Guideline 5 suggests that users should not be burdened by cognitive load and difficult learning tasks [13, 15]. If there are tasks that are not easy to learn, the designer should try to automate them as much as possible. Guideline 6 is suggested in many popular books [9, 13]. Guideline 7 is important for dealing with errors [3, 9, 11, 12, 13, 14, 15, 17]; it is related to guidelines 3 and 6 because loosely coupled systems have advantages over tightly coupled systems. It is easy to make changes to loosely coupled components. Guideline 8 makes sure that the design is expressible in multiple levels of abstraction without significant loss of clarity. When one level of abstraction is transformed into another level, consistent interpretations should be applicable to both levels. Presenting user interface designs in multiple levels may help thorough reviews. Most designers are inspired by the success of Steve Jobs and Jony Ive's design [7, 8]. Steve Jobs and Jony Ive were committed to Apple's proclamations such as "Simplicity is the ultimate sophistication" [8]. They achieved simplicity by conquering complexities, not ignoring them [8]. Jobs and Ive forged a bond that led to "the greatest industrial design collaboration of their era" [8] (page 341).

Iterative efforts outlined above were implemented in a project-based learning environment [6] with a special emphasis on design and design review. Students were introduces to the iterative efforts with an example before they began their work on their projects. Students started their project work with requirements engineering and submitted their initial requirements analysis, which was

reviewed jointly by teachers and students. In the next step, students designed the system that was followed by a design review jointly by students and teachers, followed by development of version-2 design, which was submitted by the students within a few days; this was followed by version-2 design review (jointly by students and teachers). In the next step an evolutionary prototype was implemented by the students; this prototype was then reviewed jointly by students and teachers. Based on the review, the system was redesigned again and this new design was reviewed and implemented. Another iteration of design, design review, implementation and testing was performed. Evidence from the end of course evaluation indicates that before the iterative efforts method was put into practice, the overall mean students' self-assessment of learning was 4.78 and the overall mean students' teaching evaluation was 4.73. After the practice, the learning figure rose to 4.96, and the teaching figure rose to 4.97. Evaluations by faculty panels suggest that students' performance had also improved according to outcomes based assessments [6, 16]. Iterative efforts suggested here may provide extra benefits towards fair grading based on students' participation in the project activities, rather than on written assignments, which can be generated completely or partly by AI tools such as ChatGPT, or other chat-bots [2, 10, 18]. It is becoming possible for AI tools to generate discussion board posts, write student papers based on a few prompts, and provide answers to essay questions on exams. Unacknowledged use of an AI tool such as ChatGPT to write essays, answer exam questions, write discussion board posts, or to complete many types of assignments may be considered ethically unacceptable. A few reasonable ways to adjust the course contents, delivery methods, and student evaluation in software engineering include project based learning [6], evaluation with case studies, iterative hands-on design, design reviews, and other collaborative activities. However, iterative efforts require additional time in order to complete three iterations for sizable projects, because fair assessment necessitates that adequate opportunities must be provided for student participation in all major software engineering activities [6].

5 Concluding Remarks

We have examined certain challenges of teaching software engineering and suggested opportunities for improvement based on our experiments with project-based learning [6]. Iterative efforts may offer opportunities for overcoming some of the challenges. Some extra benefits of teaching iterative efforts include engaging students in the collaborative review of all major software engineering tasks in each iteration, and outcomes based assessment for fair grading. A set of guidelines is evolving from the studies of successful design and development practices; these guidelines have great potential for helping software engineers,

and providing clarity about the nature of improvements that are achievable through iterative efforts in software engineering. From practical experience of software engineers, the guidelines can be reviewed and revised. There are opportunities for future research in this area for creating a set of revealing software engineering projects of appropriate sizes and explaining their potential benefits in project based learning environments. In addition, future work may include how to deal with ethical concerns about use of smart AI tools in teaching learning environments.

References

[1] A. Albrecht and J. Gaffney Jr. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE transactions on Software Engineering*, 9(6):639–648, 1983.

[2] A. Alkhatlan and J. Kalita. Intelligent tutoring systems: A comprehensive historical survey with recent developments. *International Journal of Computer Applications*, 181(43):1–20, 2019.

[3] E. Braude and M. Bernstein. *Software Engineering: Modern Approaches*. 2nd Ed. John Wiley, 2011.

[4] Kent Beck et al. Manifesto for agile software development. `https://agilemanifesto.org/`.

[5] J. Giacomin. What is human centered design? *The Design Journal*, 17(4), 2014.

[6] P. Guo, N. Saab, L. S. Post, and W. Admiraal. A review of project-based learning in higher education: Student outcomes and measures. *International Journal of Educational Research*, 102(4), 2020. `"https://www.sciencedirect.com/science/article/pii/S0883035519325704"`.

[7] J. Hong. Why is great design so hard? *Communications of the ACM*, 53(7,8), 2010.

[8] W. Isaacson. *Steve Jobs*. Simon and Schuster, 2011.

[9] D. Kung. *Object-Oriented Software Engineering: An Agile Unified Methodology*. McGraw-Hill, 2014.

[10] E. Mollick. ChatGPT is a tipping point for AI. *Harvard Business Review*, 2012.

[11] J. Ousterhout. *A Philosophy of Software Design*. Yaknyam Press, 2018.

[12] S. Pfleeger and J. Atlee. *Software Engineering.* Prentice-Hall, 2010.

[13] R. Pressman and B. Maxim. *Software Engineering.* 8th ed. McGraw-Hill, 2015.

[14] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual.* 2nd Ed. Addison Wesley, 2005.

[15] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 6th Ed. Pearson, 2016.

[16] Z. Sokhanvar, K. Salehi, and F. Sokhanvar. Advantages of authentic assessment for improving the learning experience and employability skills of higher education students: A systematic literature review. *Studies in Educational Evaluation*, 70, 2021.

[17] I. Sommerville. *Software Engineering.* 9th Ed. Addison Wesley, 2010.

[18] M. Welsh. The end of programming. *Communications of the ACM*, 66(1), 2023.

Bug Battles: A Competition to Catch Bugs in Different Programming Languages*

Waleed Alhumud[1], Abdullah Alenzi[1], Renée Bryce[1], Yuan Li[1] and Nasser Alshammari [2]
[1]Computer Science and Engineering
University of North Texas
Denton, TX 76203
{WaleedAlhumud,AbdullahAlenzi}@my.unt.edu,
{Renee.Bryce,Yuan.Li2}@unt.edu
[2]College of Computer and Information Sciences
Jouf University
Aljouf, Saudi Arabia
Nashamri@ju.edu.sa

Abstract

Software Engineers are typically expected to know multiple programming languages and technologies while continuing to learn as technology advances. Some students find new languages intimidating. We attempt to minimize student mindsets of intimidation when it comes to learning new programming languages while also strengthening their software testing skills through Bug Battles competition. Bug Battles is a team-based competition between students in a classroom to catch bugs in several problem sets that are written in different programming languages. In a study of 104 participants, results indicate that 93.26% of participants agree or strongly agree that Bug Battles competition improved their software testing skills and motivated them to learn new programming languages.

1 Introduction

Programming languages and technologies have changed rapidly over the years. Software Engineers are typically expected to learn new programming languages tools throughout their careers. Many students find it overwhelming and intimidating to learn new programming languages [6]. Good instructors, peers, pair programming, books, videos, and online forums often provide support to help students learn new programming languages. However, some students change majors due to different experiences related to working in teams, as well as they feel overwhelmed that it was difficult to learn one language and worry about trying to learn new languages [6][8][9]. Bug Battles strives to expose students to a mix of familiar and new programming languages in a team-based competition environment in an attempt to help students to organically notice and discuss different programming languages while searching for bugs.

Testing software is challenging as companies attempt to develop quality software within finite budgets. Low quality software may result in expensive failures or could even cause serious accidents [16]. Therefore, the expense of testing software can frequently exceed 50% or greater of the entire software development process [12]. Finding and identifying errors can be a challenging task and requires significant time and effort, thus software testing plays a crucial role in the Software Development Life Cycle (SDLC) [1]. In fact, Li and Tan reported that catching bugs can be a stimulating and inspiring task [11]. Teaching students to use software testing techniques to test software is not an easy task [10]. There are many ways to improve software testing skills. For instance, instructors may give assignments to find errors in small programs, offer help in labs, and hold competitions to catch bugs in an effort to enhance software testing skills. Working in teams is often beneficial as multiple studies, such as those on pair programming, have shown that students tend to enjoy collaborating [3].

In this paper, we present Bug Battles, a competition which is a team-based competition to catch bugs in code written in different programming languages. In Bug Battles, students compete to find bugs in code where problems are written in three different programming languages. We hypothesize that Bug Battles experiences will increase software testing skills while also motivating students to learn new languages. We survey the participants to assess whether participating in team-based competitions improves participants' software testing skills and motivates them to learn new programming languages.

The paper is organized as the following: Section 2 discusses motivation and previous work, Section 3 presents the methodology including the two surveys and the competition, Section 4 discusses results, Section 5 explains limitations steps that we took to minimize threats to validity in our study, and Section 6 gives conclusions and suggests areas of future work.

2 Background

Some studies indicate that gaining knowledge of software testing skills can help students to improve their programming skills [15]. In the software testing field, especially in the education field, there are many researchers that aim to increase students' software testing skills in different ways. Clegg et al. [5] used Code Defender which is a program that can make games between students to help them to improve their software testing skills. In the Code Defender game, students write tests like unit tests and they can evaluate existing tests to determine whether they are good or need more improvements. The game has a scoring system between the attackers who gain points by keeping their mutants survive and the defenders who gain points by writing tests that can expose those mutants. Every team tries to gain the highest scores which make the game exciting. Moreover, Fraser et al. [7] utilized Code Defenders as a graded part of the software testing course with two-academic-hour per week as practical sessions. They added new features, including the ability to evaluate each participant by using a number of analytics and statistics of his/her mutants and tests. After conducting a survey, they found that the majority of students expressed positive feedback on the integration of Code Defenders in the software testing course. In both previous studies, we share the same goal which is improving students' software testing skills. However, our study differs by providing three programming languages and these languages were selected based on students' preferences. Scatalon, Garcia, and Barbosa's research [13] was to discover the used practices to combine software testing into programming courses. They examined various research papers to identify effective methods for teaching students to improve their programming skills through the use of software testing practices. One of the successful approaches they found was holding software testing competitions between students to motivate them and increase engagement. In addition, software testing competitions can help to improve software testing skills. For instance, Bug Catcher is a web app for software testing competitions where students compete to find bugs in code as quickly as possible and report positive results from students [4]. Bug Battles differs as we use three different programming languages instead of only one. However, Bug Battles and Bug Catcher both are team-based competitions. Matthew Barr relayed on a fundamental course to prepare students to learn any new programming languages with the necessary skills [2]. Unlike traditional courses that focus on a single language, Barr utilized this course to emphasize the study of multiple programming languages. For example, students learn the differences and similarities between several programming languages. Our work differs as Bug Battles is not an entire course, but rather a team-based game that may be used in courses to try to increase student interest and enthusiasm in learning new programming languages and improving software testing skills.

3 Methodology

A competition was held and two surveys were conducted on a group of volunteers. A pre-survey was applied to determine what programming languages will be used in the Bug Battles competition. The competition was held then a post-survey was conducted to determine whether team-based competitions increase participants' software testing skills and encourage participants to learn new programming languages.

3.1 Subjects

This research was reviewed and approved by The University of North Texas Institutional Review Board (IRB). A total of 104 students who are graduates and undergraduates in the Department of Computer Science and Engineering at the University of North Texas participated in this study. The participants came from different instructors and classes because the participation in this study was volunteering.

3.2 Pre-Survey

Prior to the competition, a pre-survey was conducted to solicit participants' feedback on their interests in programming languages that they would like to see in Bug Battles. In this survey, the participants shared their opinions about the programming languages that they are most familiar with and the programming languages that they would like to be included in the competition. In the pre-survey, participants' preferences for programming languages were Python, Java, C++, C#, and Javascript.

The pre-survey results indicated that Python, Java, and C++ are the most commonly known and desired languages for inclusion in competition among the participants in Bug Battles.

3.3 Bug Battles Competition

Bug Battles which is a team-based competition was held in a classroom. The competition was held outside of class hours and it took place in eight separate sessions at varying times each day over the span of two weeks. The participants were divided into teams and each team has 4 members. Each team was given 18 questions (6 for each programming language) that contain zero to one bug. Each question includes a short problem description, a buggy code, the code's output, a field to enter the line of error, and a bug fixed field to enter the correction of error as shown in Figure 1. Each team has to find the bug in each question (if any) and write the line number of the error and how to fix it. Participants have to cover all the statements of the buggy code to detect any

#6. This code prints names in the list except the name that was passed to the function

```
def names_filter(given_name):
    names = ["Larry", "Mark", "Cris", "Lebron", "Tomas"]
    for name in names:
        if name == given_name:
            break
        else:
            print(name)

names_filter("Mark")
```

Output should be	Larry Cris Lebron Tomas
Line of Error	
Bug fixed	

Figure 1: Question sample.

errors. In case there is no error or participants do not know the answer, they write "0" in the line number of error and "Do not know" in the bug fixed field. Each team receives points for the number of bugs that they catch and fix.

3.4 Post-survey

The post-survey was conducted after the Bug Battles competition to collect individual opinions about the Bug Battles experience. The survey is based on a five-point likert scale that captures respondents' level of agreement: Strongly disagree, Disagree, Neutral, Agree, and Strongly agree [14]. The main goal of the post-survey is to answer the following research questions based on the participants' responses:

- RQ1: Do Bug Battles increase participants' software testing skills?
- RQ2: Do Bug Battles encourage participants to learn new programming languages?

4 Results

The post-survey results indicated that more than 93% of the participants strongly agreed or agreed that detecting bugs with different programming languages enhances their software testing skills. On the other hand, less than

3% of the participants strongly disagreed or disagreed while less than 4% were neutral as shown in Figure 2. It is important to note that these positive responses prove how competitions might improve software testing skills. These findings are consistent with previous research in terms of improving software testing skills after experiencing the competition [4].

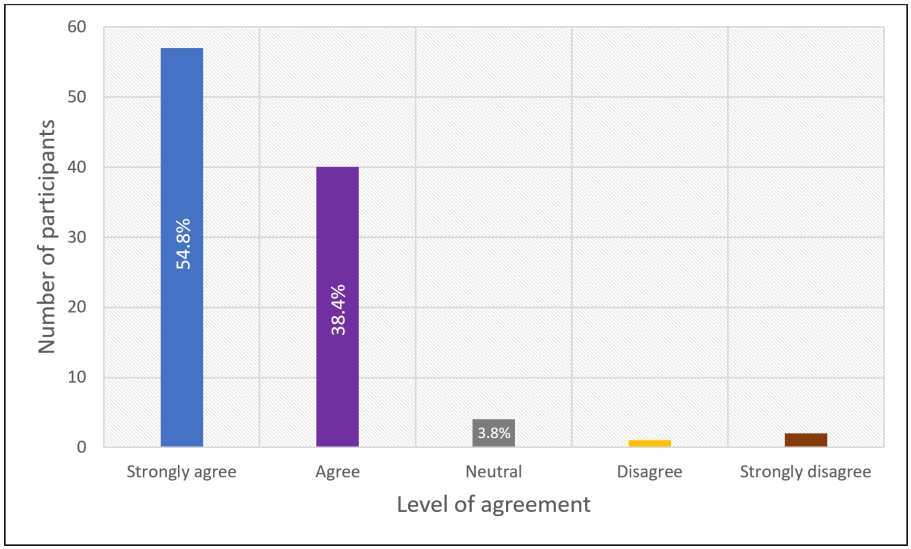


Figure 2: Result of increasing software testing skills.

Figure 3 presents that 93.26% of participants agreed or strongly agreed that competition like this motivates them to learn more programming languages. This result is not surprising to us since some students were new to one or more languages in the competition. Another explanation for this result is using different programming languages in the competition and these languages were included based on their preferences. It is noteworthy that none of the participants disagreed, while only 2 expressed strong disagreement.

The vast majority of the participants believed that their skills were improved to compare programming languages' syntax. Some participants did not know that they do not need to declare the type of variables in some languages like Python, as they do in Java or C++. The Python problems set helped them to discover the nature of the dynamic programming language. More than 80% of the participants preferred working in teams rather than working individually to catch bugs. In contrast, a tiny minority of the participants who disagreed or strongly disagreed to work in teams with approximately less than 2%. The participants shared their points of view regarding the excitement of Bug Battles

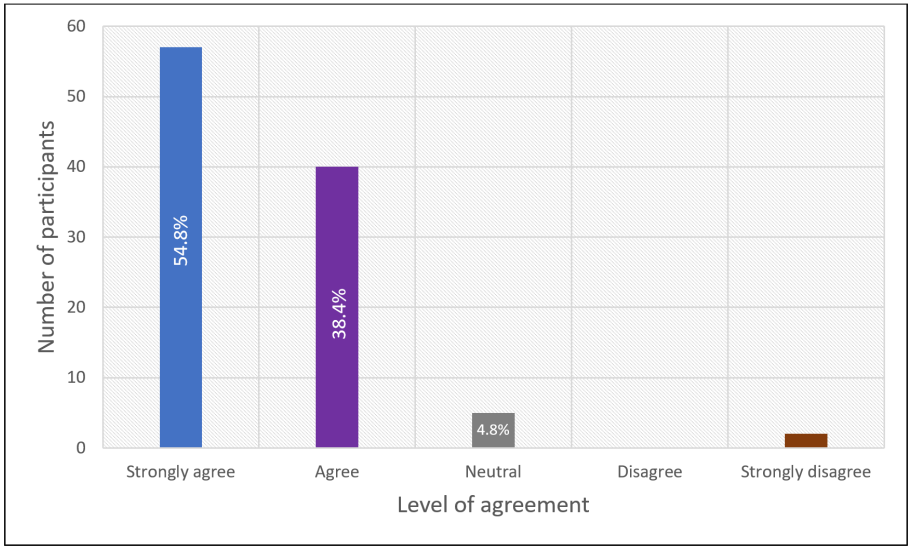
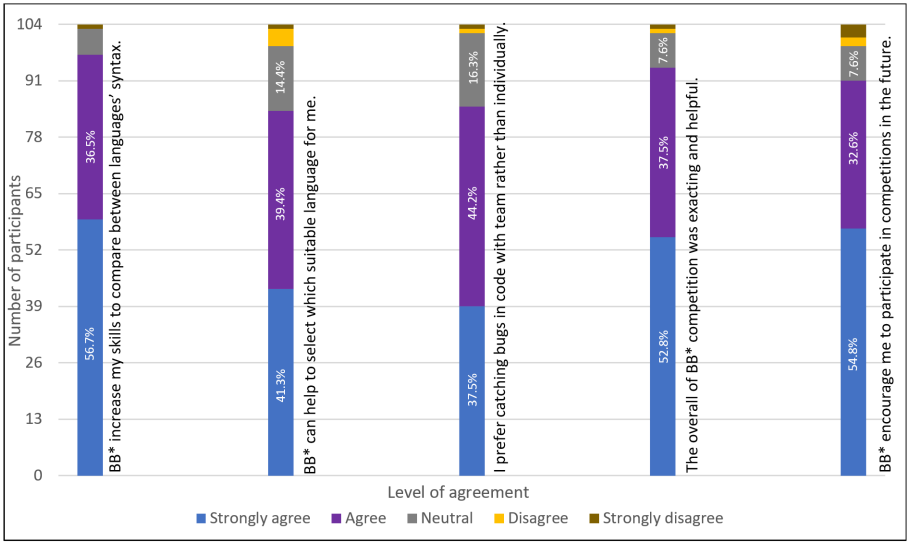


Figure 3: Result of increasing motivation to learn new programming languages.

and whether it motivated them to join future bug-catching competitions. The results showed that a high percentage of the participants found Bug Battles to be thrilling and beneficial. Additionally, 87.5% of the participants either agreed or strongly agreed that Bug Battles had inspired them to participate in finding bugs competitions in the future. Figure 4 shows all the results in this paragraph.

The findings of our research have implications for instructors as they can be used to improve their students' software testing skills and motivate them to learn new languages by holding team-based competitions. Additionally, these findings can have implications for software engineers as they can provide an opportunity for engineers to learn new languages and practice finding bugs. When participating in team-based competitions, engineers may encounter bugs and errors that they have never encountered before, which can provide a valuable learning opportunity to expand their software testing skill set. Moreover, in team-based competitions, engineers may have a learning experience as they communicate and collaborate with their team members in order to find and fix bugs.



Note: *BB = Bug Battles

Figure 4: Results of the other post-survey questions.

5 Threats to Validity

There are factors in our study that prevent us from generalizing the results to all students. We tried to minimize this risk by reaching 104 students. Another threat to validity is that different coding problems and bugs may change the results. We tried to minimize this risk by using 18 problems in the competition (six per programming language) and including bugs that we categorized as beginner, intermediate, and advanced in order to engage students.

6 Conclusion and Future Work

Bug Battles is a team based competition in which teams compete to catch the most bugs in problem sets written in Python, Java, and C++. These languages were selected by the students in a pre-survey in an effort to include both familiar and new languages for the students. The results of the survey demonstrate that students have positive feelings toward Bug Battles in terms of motivating them to learn new languages and improving their software testing skills. We recommend instructors of computer programming courses organize team-based competitions to assist students in enhancing their software testing skills and inspire them to learn new programming languages. Problem sets and

solutions may be freely downloaded from GitHub:
`https://github.com/Waleed9549/bugBattles`

Future work will examine Bug Battles competitions with more programming languages and problem sets. We plan to conduct a student focus group to develop problem sets based on bugs that focus group participants have encountered in different programming languages. We plan to solicit feedback from instructors and teach assistants about bugs that they notice their students encounter in an effort for us to include common bugs in Bug Battles. We hypothesize that customizing competitions to issues encountered by students will help students succeed in our programs.

References

[1] Abdullah Alenzi, Waleed Alhumud, Renée Bryce, and Nasser Alshammari. A survey of software testing tools in the web development domain. *J. Comput. Sci. Coll.*, 38(2):63–73, dec 2022.

[2] Matthew Barr. How to learn a new language: A novel introductory programming course. In *Computing Education Practice*, pages 9–12. 2023.

[3] Renee Bryce. Bug wars: A competitive exercise to find bugs in code. *J. Comput. Sci. Coll.*, 27(2):43–50, dec 2011.

[4] Renée Bryce, Quentin Mayo, Aaron Andrews, Daniel Bokser, Michael Burton, Chelynn Day, Jessica Gonzolez, and Tara Noble. Bug catcher: A system for software testing competitions. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 513–518, 2013.

[5] Benjamin S Clegg, José Miguel Rojas, and Gordon Fraser. Teaching software testing concepts using a mutation testing game. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 33–36. IEEE, 2017.

[6] Paul Denny, Brett A Becker, Nigel Bosch, James Prather, Brent Reeves, and Jacqueline Whalley. Novice reflections during the transition to a new programming language. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 948–954, 2022.

[7] Gordon Fraser, Alessio Gambi, and José Miguel Rojas. Teaching software testing with the code defenders testing game: Experiences and improvements. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 461–464. IEEE, 2020.

[8] Patricia Haden, Dale Parsons, Krissi Wood, and Joy Gasson. Student affect in cs1: Insights from an easy data collection tool. Koli Calling '17, page 40–49, New York, NY, USA, 2017. Association for Computing Machinery.

[9] Amanpreet Kapoor and Christina Gardner-McCune. Considerations for switching: exploring factors behind cs students' desire to leave a cs major. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 290–295, 2018.

[10] Per Lauvås and Andrea Arcuri. Recent trends in software testing education: A systematic literature review. In *NIK*, 2018.

[11] Ziqiang Li and Shin Hwei Tan. Bugine: a bug report recommendation system for android apps. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 278–279, 2020.

[12] Congcong Liu. Research on software test data generation based on particle swarm optimization algorithm. In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1375–1378, 2021.

[13] Lilian Passos Scatalon, Rogério Eduardo Garcia, and Ellen Francine Barbosa. Teaching practices of software testing in programming education. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2020.

[14] Bert Weijters, Kobe Millet, and Elke Cabooter. Extremity in horizontal and vertical likert scale format responses. some evidence on how visual distance between response categories influences extreme responding. *International journal of research in marketing*, 38(1):85–103, 2021.

[15] Jacqueline L. Whalley and Anne Philpott. A unit testing approach to building novice programmers' skills and confidence. In *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114*, ACE '11, page 113–118, AUS, 2011. Australian Computer Society, Inc.

[16] Yi Zhao, Yun Hu, and Jiayu Gong. Research on international standardization of software quality and software testing. In *2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall)*, pages 56–62, 2021.

Preparing ABET Accreditation for An Undergraduate Software Engineering Program*

Jicheng Fu, Myungah Park, Gang Qian,
Hong Sung, Thomas Turner
Department of Computer Science
University of Central Oklahoma
Edmond, OK 73034
gqian@uco.edu

Abstract

In this article, we share our seven-year experience in preparing for ABET accreditation for the undergraduate Software Engineering program offered at the University of Central Oklahoma. We present our approach to the preparation with a focused discussion of the general and the program-specific accreditation criteria required by ABET Engineering Accreditation Commission. As most computer science graduates work in the area of software development, we hope our experience can benefit computer science departments similar to our setup and promote better undergraduate education in the area of software engineering.

1 Introduction

The BS in Software Engineering (SE) program was created at the University of Central Oklahoma (UCO) in 2014. Before that, the department had been offering a BS in Computer Science program for almost thirty years. Our motivation to create the SE program was multifold. First, while SE used to be considered as a sub-discipline of Computer Science (CS), software systems had grown larger, more complex, and more expensive, some of which were even

mission critical. As a result, SE was eventually recognized as a discipline of its own by many professionals in the IT industry. Second, it was our observation that most of our CS graduates worked in positions related to software design and development. Based on feedback from our alumni and the department's industrial advisory board, offering a program that focused on the lifecycle of software systems would greatly benefit our graduates for their career in the software industry.

After the BS in SE program was created, an immediate next step for us was to prepare for ABET accreditation. The department received ABET accreditation for its CS major in 2007, of which we recognized the benefits as follows: 1) it increased the recognition of the program from the university administration; 2) it increased the recognition of the program from its external stakeholders, especially prospective students; and 3) it helped maintain an objective academic standard that was established by third-party experts in the CS discipline. Our experience of having the CS major accredited was documented in [4] and [3].

To accredit the new SE program, we worked with a different ABET Commission. While all computing disciplines are reviewed and accredited by ABET Computing Accreditation Commission (CAC), all engineering programs are evaluated by ABET Engineering Accreditation Commission (EAC). Therefore, we studied and implemented ABET standards for engineering programs before the ABET EAC review of 2019. The remainder of the paper discusses the details of our approach to preparing for ABET accreditation with a focus on both the general and the SE program-specific accreditation criteria.

2 Preparing for the ABET Evaluation

Our preparation for the ABET evaluation of our SE program included the following features:

1. Having an existing program accredited by ABET
2. Having a member of the faculty serve as an ABET volunteer
3. Performing a dress rehearsal for the SE program accreditation visit

While none of features above were mandatory, each of them provided support in the accreditation process. Having an existing program accredited by ABET meant that the department had a working infrastructure for documentation to support ABET accreditation. As indicated by Leach [5], documentation is important; if it is not written, it does not exist. The department had a process for establishing and reviewing program educational objectives (see Section 3.2). We also had functioning assessment tools and procedures to enable continuous improvement. A method for collecting required documents for courses and storing them electronically was instituted too. The sum of these activities served as a foundation for accrediting the SE program.

Having a member of the department serve as an ABET volunteer provided detailed insights as to how ABET accreditation was performed: this knowledge was invaluable in preparing for ABET visits. The institution and the department recognized the effort of the ABET volunteer, demonstrating support for the ABET accreditation process as recommended by Collofello [2].

Based on prior experiences, the department believed that a dress rehearsal for a future ABET evaluation would be best. We contacted the ABET Foundation, now called the ABET Bridge, and asked for a former team chair whose area of expertise was in the area of SE. The ABET Foundation found a consultant having the appropriate credentials, who reviewed a self-study that we prepared and visited our institution to conduct a review exactly as an ABET team would do. Based on the consultant's observations, we revised the assessment processes and the materials taught in some classes to conform to the consultant's suggestions.

3 ABET Criteria for Software Engineering Programs

In this section, we present in detail how we met the ABET criteria for Software Engineering programs, which must be followed by every accredited SE program. As mentioned earlier, ABET EAC accredits Software Engineering programs instead of ABET CAC. The 2022-23 Criteria can be found at [1]. The Criteria are divided into the General Criteria for Baccalaureate Level Programs, General Criteria for Master's Level and Integrated Baccalaureate – Master's Level Engineering Programs, and Program Criteria. In this paper, we limit our discussion to the baccalaureate-level General Criteria as well as the Program Criteria for SE programs. ABET evaluates programs to ensure that they satisfy both the General and the Program Criteria.

There are eight General Criteria and two Program Criteria for SE programs. The Criteria are discussed in each of the following subsections. The two SE Program Criteria are related to curriculum and faculty so they are discussed in the related General Criteria subsections (sections 3.5 and 3.6, respectively).

3.1 Criterion 1. Students

Criterion 1 is about student admission, performance evaluation, transfers, advisement, and graduation requirements. This criterion ensures that students are advised regarding curriculum and career matters, attain all outcomes by the time of graduation, are awarded appropriate academic credits for all courses, and satisfy all graduation requirements.

As mentioned in the previous paragraph, student advisement is a major concern of this criterion. When we obtained ABET accreditation for our CS major, the faculty advised every student every semester. The workload became

unsustainable as our enrollment more than doubled in the past ten years. As a result, we reformed our advisement process. Each semester, we only offer advisement to new students to the department (transfers, major changes, and freshmen) as well as those who try to enroll into a few key courses in the curriculum, including Programming II, Data Structures & Algorithms, and the SE Senior Design. We used a combination of enrollment holds and instructor permissions to enforce the mandatory advisement process. Each advisement session was documented. This approach reduced our workload by half.

We complied with the admission requirements set forth by the university to accept new students and transfer students. There were no additional admission requirements for students who chose to pursue the SE degree. Student performance was evaluated based on the student outcomes that ABET specifies in Criterion 3 (see Section 3.3). It was the department's responsibility to create and administer assessment instruments to evaluate student outcomes.

Another concern of this criterion was awarding credits to transfer students. In this case, we handled the SE program in the same way as we handled our ABET-accredited CS major. In general, we required that all upper-division SE and CS courses be taken at UCO. The details can be found in [3].

3.2 Criterion 2. Program Educational Objectives

ABET EAC shares the same definition of Program Educational Objectives (PEOs) as that of ABET CAC, which are "... broad statements that describe what graduates are expected to attain within a few years after graduation". It is important to note that the specified attainments in the PEOs should not be acquirable by a student before his/her graduation. The PEOs should only be obtained after the student graduates.

Criterion 2 requires that PEOs be consistent with the mission of the institution. In addition, a review process of the PEOs, which involves constituencies of the program, must be conducted and documented with a regular cycle [3]. We ensured that every PEO was directly supported by one or more student outcomes (see Section 3.3) to satisfy General Criterion 3. The department published the PEOs on the department website as required by ABET.

3.3 Criterion 3. Student Outcomes

Student Outcomes (SOs) describe "what students are expected to know and be able to do by the time of graduation". There are seven ABET-defined SOs under Criterion 3, which describe the knowledge, skills, and behaviors that students should acquire through the program. We directly adopted all seven ABET SOs as the SOs of the SE program. Note that SOs must be periodically reviewed, and a program can choose to define its own SOs [3].

Every SO needs to be related to its corresponding PEOs. In our self-study report for accreditation, we utilized a table format to illustrate how each PEO was supported by the SOs. For example, as shown in Table 1, PEO (b) was supported by SOs #3, #4, #5, #6, and #7.

Table 1: Mapping of SOs to PEOs

PEOs	SOs						
	#1	#2	#3	#4	#5	#6	#7
(a)	x	x	x	x	x	x	x
(b)			x	x	x	x	x
(c)	x					x	x

3.4 Criterion 4. Continuous Improvement

Criterion 4 requires that students must be assessed to evaluate the extent to which SOs are being attained on a regular basis. ABET defines Assessment as "one or more processes that identify, collect, and prepare data to evaluate the attainment of student outcomes". It defines Evaluation as "one or more processes for interpreting the data and evidence accumulated through assessment processes".

Criterion 4 was the most time consuming and difficult criterion for us to satisfy. As a first step, we needed to establish the assessment instruments. For our ABET-accredited CS major, we used to rely on a locally defined test as the main instrument [3]. This approach worked for ABET accreditation purposes. However, we found that it was difficult for us to fully utilize the assessment/evaluation results for continuous improvement. Therefore, we decided to switch to a course-oriented approach.

For each SO, we started by identifying all the courses where students were being prepared to attain that SO. Among the courses identified, we picked one course that was a required course for the SE program, in which the specific SO would be assessed. The next step was to choose one or more performance criteria (PCs) to evaluate the SO. The number of PCs to use can be determined based on the complexity of the SO. For simple SOs, a single PC is sufficient. For complex ones, two or even more PCs may be needed. For example, for SO#6, "an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions", we defined two PCs as follows:

- PC#1 for SO#6: An ability to development and conduct appropriate experimentation

- PC#2 for SO#6: An ability to analyze and interpret data, and use engineering judgment to draw conclusions

Once the PCs are settled for an SO, artifacts produced by students in the chosen course should be used as the assessment instrument for each PC. Note that ABET requires direct measures be used in this process. Therefore, we needed to use actual work produced by students (indirect measures such as opinion surveys would not count). Depending on the nature of the PC, a homework assignment, a course project, or a test can be used from the chosen course. Note that an assessment instrument need not to be the entire assignment or test. A part of an assignment or test can also serve as an assessment instrument. For each artifact gathered, we defined a rubric to evaluate the artifact. For example, an artifact might be rated in one of the following categories: Unsatisfactory, Developing, Satisfactory, or Excellent.

Figure 1 provides an illustration of all the assessment items discussed above. As shown in the figure, the course SE 4213 was chosen for the assessment of SO#2. SO#2 had two PCs. PC#1 used the course project as its assessment instrument while PC#2 used the second test of the course as its assessment instrument.

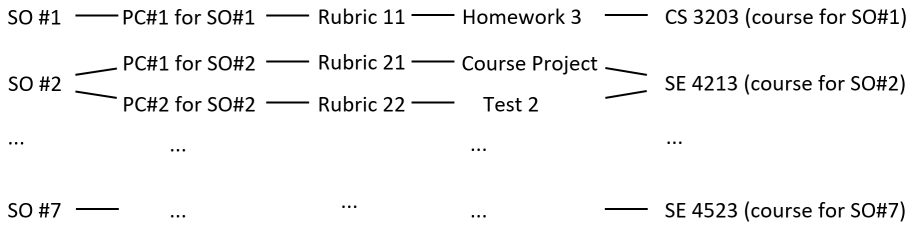


Figure 1: Illustration of Relationships of the Assessment Items

After the course-oriented assessment structure was established, we set up an evaluation schedule since ABET requires continuous assessment cycles. In the schedule, artifacts gathered in the spring and fall semesters of the prior calendar year were evaluated in the spring of the subsequent calendar year. For example, artifacts gathered in the spring and fall semesters of 2018 were evaluated in the spring of 2019. Annual assessment reports were produced to document the results of evaluating all SOs. In the annual assessment reports, we identified opportunities for improving the SE program. The department's assessment committee decided upon which of these opportunities the department would act, based on resources available and the effort required to address the opportunity.

3.5 Criterion 5. Curriculum

We must meet both the General Criterion and the SE Program Criterion on Curriculum as set by ABET. The General Criterion specifies the subject area and the number of credit hours required for mathematics, basic science, and engineering topics without naming specific courses. The Program Criterion mandates SE-related topics such as computing fundamentals, discrete math, software design and construction, requirements analysis, security, and software verification and validation.

We ensured that every required SE topic was covered and every SO was supported by some course(s) in the curriculum. As mentioned earlier, the department offered the SE program along with an ABET-accredited CS major. Many of the courses were shared between the two degrees. Several lower-division courses were characterized to meet both the CS and the SE curricula for ABET accreditation.

At the upper-division level, three dedicated SE courses were created along with a capstone course for senior design. The program offered three application areas, including application development, system development, and a newly added cybersecurity track. Students in the SE program could take CS courses as electives based on the application area that they chose.

This criterion also requires that course collectible items be kept for each time a course is taught. In addition, a program needs at least one graduate under an ABET-compliant curriculum to qualify for an onsite visitation for accreditation [3].

3.6 Criterion 6. Faculty

General Criterion 6 covers faculty size, expertise and experience while the SE Program Criterion on Faculty requires that the core SE faculty members maintain currency in their areas of professional or scholarly specialization. At the time when we applied for ABET accreditation, there were nine full-time faculty members in the department. Among them, seven held a Ph.D. and two held an MS degree. Most of the faculty members had also worked in the industry on software development.

ABET requires that "collectively, the faculty must have the breadth and depth to cover all curricular areas of the program." To meet this requirement, two core faculty members were primarily responsible for teaching SE-related courses while other faculty members taught the rest of the required and elective courses.

Similar to the situation of many other regional universities that focused on teaching excellence, the most challenging requirement of this criterion for our department was to demonstrate that the faculty members had ongoing

professional development to maintain currency. Fortunately, we were able to adopt the same approach that we used for the accreditation of our CS major. The actual approach was discussed in [3].

3.7 Criterion 7. Facilities

Criterion 7 examines the resources that are available to the program, such as equipment, classrooms, and laboratories. For this criterion, we ensured that all lab computers were upgraded on a fixed five-year cycle. Three faculty members served as system administrators for the department's computer servers to promptly resolve any issues that might affect students' course works.

3.8 Criterion 8. Institutional Support

Criterion 8 emphasizes the importance of institutional support and leadership that must be "adequate to ensure the quality and continuity of the program". Our accreditation effort was well supported by the administration at both the university and the college levels. In general, administration support is indispensable for a program to obtain ABET accreditation.

Our two ABET-accredited programs (SE and CS) shared the same approaches to satisfy criteria 7 and 8. To avoid repeating ourselves, we refer our readers to [3] for more details.

4 Conclusion

In this article, we presented our experience in preparing for ABET accreditation for our undergraduate SE program. A timeline of the major events is presented as follows:

1. Fall 2014: The BS in SE program was created.
2. Spring 2016: The first graduate of the program received his degree.
3. Summer 2016: An ABET consultant visited us and provided his evaluation and feedback.
4. Fall 2019: A team of ABET evaluators conducted onsite visitation.
5. Summer 2020: The program was officially accredited by ABET EAC.

Currently, we have about 260 students in the ABET-accredited CS major and about 70 students in the SE program. We have observed a noticeable enrollment increase in the SE program after its ABET accreditation. The accreditation of the SE program has also led to its top national rankings by two separate online resources for prospective college students (study.com and HQUniversity). For the next step, we will be working on ways to encourage even more students into the SE program as it provides targeted learning on

software design and development, which is a much-needed skill for the professional career of most of our graduates in the industry.

References

[1] ABET. Engineering accreditation commission, 2022–23 criteria. `https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2022-2023/`.

[2] James Collofello. Applying lessons learned from software process assessments to ABET accreditation. In *34th Annual Frontiers in Education, 2004. FIE 2004.*, pages T3G–24. IEEE, 2004.

[3] Jicheng Fu, M. Gourley, M. Park, G. Qian, H. Sung, and T. Turner. Obtaining and maintaining ABET accreditation: An experience-based review of the ABET criteria for computer science programs. *Journal of Computing Sciences in Colleges*, 29(4):13–19, 2014.

[4] Michael Gourley, G. Qian, H. Sung, and T. Turner. Seeking ABET accreditation of a computer science program at a public regional university. *Journal of Computing Sciences in Colleges*, 23(6):140–147, 2008.

[5] Ronald J Leach. Using the vocabulary of software engineering to describe ABET accreditation. *ACM Inroads*, 1(2):27–29, 2010.

Comparative Sequential and Parallel Discrete Signal Convolution Algorithms: A Case Study*

Caleb Sneath and Eduardo Colmenares
Computer Science
Midwestern State University
{caleb.sneath, eduardo.colmenares}@msutexas.edu

Abstract

Case studies play an important role in the testing and verification of prevailing conventional wisdom, the collection of knowledge in a new and quick format, and to potentially answer several minor questions at once which on their own are not worthy of deeper inquiry. Case studies ideally should have a clear focus, provide at least one good question which can be answered by the case study, and ensure that their creation provides some tangible educational benefit to one or more fields. This paper focuses on discrete signal convolution problems, as well as multiple algorithms which can be used to solve them, particularly from the viewpoint of output signal analysis.

The discrete one-dimensional (1D) signal convolution problem has applications in statistics, physics, music, machine learning, and electrical engineering. This paper aims to answer how much a parallel GPU-based approach might speed up the problem as well as ways an existing library implementation can differ from a simpler parallel algorithm implementation from scratch. This paper presents a case study that makes extensive use of GPU programming and should benefit not only those involved in computer science, but also those involved in any mathematics, engineering, or fields of science in which signal convolution is a concern.

1 Methodology

This paper explores 1D convolution by breaking its discussion into six parts. The first part aims to provide a quick definition and summary of the relevant background information for convolution. The second step goes over one possible method to solve 1D convolution problems by hand. This section also contains an illustrative example. The third part discusses ways in which the 1D convolution problem can be programmed. This section describes two methods to view 1D convolution problems that help break it down easier into sequential code, a section on how to modify one of the sequential algorithms to instead take advantage of GPU parallelization, and a section which briefly mentions some modifications which a more complicated library might use in order to further optimize the problem. The sections on sequential and parallel code each include example C++/CUDA implementations of 1D convolution algorithms. Finally, the last part of this paper summarizes an experiment to time one of the sequential approaches as well as the parallel GPU approach mentioned in in part three of this paper. This section includes tables of the results of the average execution times and speedups across different problem sizes. The averages for each input size were obtained from 35 trials each of input matrix sizes in increasing powers of two. For each trial, the filter size remained constant with a size of four, and each item of the output matrix was computed by a single GPU thread each for the parallel trials. This section also briefly lists the hardware specifications of the test machine used, although all trials were conducted on the same computing cluster.

2 Problem Description

Convolution is an operation like addition or subtraction, but for matrices rather than individual numbers. It is represented by the "*" symbol, meaning that for matrices it is important to properly distinguish between multiplication and convolution. For this reason, normal multiplication should be represented by "x", and the dot product by a single dot. As an operation, convolution is commutative so the order of the two matrices to be convolved does not matter [5][4]. The larger matrix is often referred to as the input matrix or mathematically as x[n]. The smaller matrix on the other hand has a variety of commonly used terms to refer to it such as filter, kernel, convolution kernel, the point spread function, or mathematically as h[n] [5].

Standard conventions on the format of the two signal matrices to be convolved exist which make communicating and calculating 1D convolution easier. One of the matrices to be convolved should be the impulse response for a given delta function. The delta function, denoted by δ[n], is a way to represent any

signal in a normalized format by giving sample number zero the value of one, and all other signals a value of zero. The impulse response then is the resulting exiting symbol for the delta function] [5][4]. In simple terms, the 1D convolution is a sum of products in a sliding window. For two 1D matrices, matrix h and matrix x, the convolution is formally defined by the following equation:

$$y[i] = \sum_{(j=0)}^{(M-1)} h[j]x[i-j]$$

Equation 1.

3 The Do It by Hand Phase

Signal convolution can be solved fairly simply by hand, although the time required to do so can be immense as the problem size expands past just a few values for the input and filter. For simplicity, this section will refer to the signal with more samples as the input signal, and the smaller signal as the filter. This should not pose a problem in practice as convolution's commutative property means that the order of the two signals can be freely rearranged without affecting the output [5].

One helpful method is to start by finding the total number of outputs for the output signal. This will simply be the sum of the number of elements in the input and filter signal minus one. Then, create a table with as many columns as the number of outputs in the output signal, and as many rows as the number of inputs. In the first row, starting in the first column and going right a column one number at a time, write each element of the filter, stopping when the filter is out of elements. Then, next to each of the newly added numbers, write a multiplication sign, and put the first element of the input next to each of these numbers. Next, move onto the next row. Once again, start by writing out all elements of the filter, however begin doing so one column to the right of the previous row. Next, once again place the multiplication sign next to each element, and this time add element two of the input signal. Repeat this process until there are no more rows. Next, compute the product for each cell. Finally, sum up every element within the same column, and left to right each sum is that element of the output signal.

For illustrative purposes let's assume signal S1 is the input signal, signal S2 is the filter signal, and signal S3 is the output signal. Although this example uses signals of size 5 and 3, it is important to remember that by following the same process, any size of inputs can be solved in a similar fashion. If S1 = 1, 2, 3, 4, 5; S2 = 5, 6, 7; the objective is then to compute S3 = S1 * S2.

```
S3 =
{ 1 x 5    , 1 x 6    , 1 x 7
+          2 x 5    , 2 x 6    , 2 x 7
+                   3 X 5    , 3 x 6    , 3 X 7
+                            4 X 5    , 4 X 6    , 4 X 7
+                                     5 X 5    , 5 X 6    , 5 X 7
}
=
{
  5        , 6        , 7
+          10       , 12       , 14
+                   15       , 18       , 21
+                            20       , 24       , 28
+                                     25       , 30       , 35
}
= { 5, 16, 34, 52, 70, 58, 35 } = S3
```

Figure 1: By Hand Example

4 Computational Solutions

There are at least two different broad approaches to creating an algorithm that can solve signal convolution problems. The first broad viewpoint involves examining and therefore iterating through each input signal to determine how it contributes to each output. This is arguably better suited to a sequential approach than a parallel one as each iteration of the outer loop contributes to multiple output signals, resulting in some dependencies on multiple iterations for each output. The second viewpoint involves examining and therefore iterating across each output signal and determining how it was affected by different inputs. This algorithm is well suited to parallelization as each iteration of the outer loop can be turned into a separate thread since each element of the output can be calculated entirely independent of the others, although it is also fine for serial approaches.

4.1 First Sequential Implementation

The input viewpoint serial code is described by Smith[5] as a very intuitive algorithm to understand. Simply put, after zeroing the output signal's array there is an outer loop which iterates from the starting element to the ending element of the input signal's array. Then, there is an inner loop which iterates from the starting element to the ending element of the filter signal. Inside the

nested loop, the output signal with an index equal to both loop counters is assigned the value of itself plus (the current input element times the current filter element). Figure 2 provides an idea of its C++ implementation.

```
void convolutionSerial(float* inArray, int inSize, float* filterArray, int filterSize, float*
returnArray)
{
	// Determine job range
	int totalSize = inSize + filterSize - 1;

	// Zero the output array
	for (int index = 0; index < totalSize; index++)
	{
		returnArray[index] = 0;
	}

	// Perform convolution in the input signal style
	// Iterate through each element of input array (x[n])
	for (int index = 0; index < inSize; index++)
	{
		// Iterate through each element of filter (h[n])
		for (int inIndex = 0; inIndex < filterSize; inIndex++)
		{
			returnArray[index + inIndex] += (filterArray[inIndex] * inArray[index]);
		}
	}
```

Figure 2: Input Viewpoint Sequential Code

4.2 Second Sequential Implementation

The output viewpoint is mainly more difficult to understand due to the complexity at first glance of determining which indices of the input and filter arrays correspond to which output without directly iterating through each to find out. It begins with an outer loop which iterates across each element of the output matrix. At the start of each iteration of the loop, the current output being calculated is assigned a starting value of zero. Then, an inner loop begins which iterates through each element of the filter signal array. Inside this loop, some bounds checking is done to ensure the input and filters don't go out of bounds for their index. If they won't, an extra step must be taken. A temporary value will be computed using the product of one point of the input and filter matrix. The filter value's index can be obtained from the innermost loop. The input matrix item for this will be the item with the index of the current output value minus the index of the current filter value. Afterwards, this temporary value is added to the current output value. An example of the use of this temporary value is below.

Input Matrix = {0, 2, 4, 6, 5} Filter Matrix = {1, 2, 3, 5}

To calculate the temporary value for the output matrix on index 3 for the innerloop value of 1, first the filter value must be obtained. The filter matrix

index is obtained directly from the innerloop index, which is 1. Filter Matrix [0] is 1. Next, the input matrix value must be obtained. The input matrix index is obtained from subtracting the current output value index, 3, from the current filter value index we just obtained, 0, for an index of 3. Input Matrix [3] is 6. Calculating the product of the input and filter gives a temporary value of (6 * 1 = 6). This temporary value can then be added to the current output value, Output Matrix [3]. Figure 3 provides an idea of its C++ implementation.

```
void convolutionSerial(float* inArray, int inSize, float* filterArray, int filterSize, float*
returnArray)
{
        // Determine output range
        int totalSize = inSize + filterSize - 1;

        // Perform convolution in the output signal analysis style
        // Iterate through each element of output array (y[n])
        // and add every element that contributes to it.
        for (int outIndex = 0; outIndex < totalSize; outIndex++)
        {
                returnArray[outIndex] = 0;

                for (int index = 0; index < filterSize; index++)
                {
                        if ((outIndex >= index) && ((outIndex - index) < inSize))
                        {
                                returnArray[outIndex] += filterArray[index] * inArray[outIndex -
index]);
                        }
                }
        }
}
```

Figure 3: Output Viewpoint Sequential Code

4.3 Parallel Implementation

This paper's illustrative parallel approach was similar to the serial output viewpoint centered algorithm with several minor adjustments. Each thread is mostly concerned with calculating one output value, however each thread may need to calculate more if it is assigned more outputs than the number of threads called, or less, meaning zero, in cases where the number of outputs isn't evenly divisible by the number of threads per block. To meet these goals, the function begins like the sequential output viewpoint version by calculating the number of total threads as well as which job id this block and thread would translate into. Then, the outer loop is tweaked to begin only at this thread's job id as the output index, and go up by the total number of threads each time until it is at least as large as the total number of outputs.

Figure 4 provides an idea of its C++ implementation.

```
__global__
void convolutionParallelKernel(float* inArray, int inSize, float*
        filterArray, int filterSize, float* returnArray)
{
        // Determine job range
        int totalThreads = gridDim.x * blockDim.x;
        int totalSize = inSize + filterSize - 1;
        int thisIndex = (blockIdx.x * blockDim.x) + threadIdx.x;

        // Iterate through all items that contribute to this assigned
        // output signal index
        for(int outIndex = thisIndex; outIndex < totalSize; outIndex += totalThreads)
        {
                returnArray[outIndex] = 0;

                for (int index = 0; index < filterSize; index++)
                {
                        if ((outIndex >= index) && ((outIndex - index) < inSize))
                        {
                              returnArray[outIndex] += (filterArray[index] * inArray[outIndex -
index]);
                        }
                }
        }
}
```

Figure 4: Parallel CUDA GPU Code

5 Testing Environment

All trials were performed in the same program execution on the Maverick2 computing cluster. The specifications of the Maverick2 cluster as presented by TACC are as follows [2][3][1].

6 Comparisons

Testing revealed several predictable as well as unexpected results. As is somewhat expected, serial time grew roughly linearly for the serial approach with respect to input size. In contrast, the parallel approach started off performing weaker, but scaled significantly better as the problem size increased in comparison, with the graph showing somewhere from constant to logarithmic performance. This results in the serial program being more efficient for smaller problem sizes, but the parallel approach gaining an advantage as the problem size increases, suggesting that it may have a better algorithmic complexity. Allocating variables, copying variables, launching multiple threads, and then recopying and freeing extra variables can constitute a relatively expensive process, so it isn't unreasonable that for small enough problem sizes, this extra overhead is simply not a worthwhile cost to solve the problem better.

One trend which warrants further investigation is the difference in how required time changes between the smaller and the larger parallel GPU based approaches. While the small input size figures produce similar averages seem-

Table 1: Maverick 2 GTX Compute Node Specifications

CPU (Intel(R) Xeon(R) CPU E5-2620 v4)			**GPU** (GTX 1080-TI)	
Processors per node:	2		CUDA Cores	3584
Cores per processor:	8		Graphics Clock (MHz)	1480
Cores per node:	16		Processor Clock (MHz)	1582
HW threads per core:	2		Graphics Performance	High-200,000
HW threads per node:	32		**Memory Specs**	
Clock rate:	2.10 GHz		Standard Memory Configuration	11 GB GDDR5X
RAM:	128 GB		Memory Interface Width	352-bit
L1/L2/L3 Cache:	512KiB/ 2MiB/ 20MiB		Bandwidth (GB/Sec)	11 Gbps
Local Storage:	150.0 GB (60 GB free)		**Thermal and Power Specs**	
GPUs:	4xNVidia 1080-TI GPUs		Max. Graphics Card Power (W)	250

ingly constant and within the normal range of variance, larger input sizes begin to scale to some degree with input sizes. This doesn't appear to be a mere outlier of the data, as similar figures were obtained across several 35-trial average runs of the program. As no thread or output should be reliant upon the work of another thread, it is possible that what is happening is that at some point, the threads simply start competing for placing required values in limited memory space. If this is indeed the case, it might be worthwhile to examine whether for a large enough problem size, factors like competition for limited memory begins to lower the relative advantage that parallel GPU approaches possess vs serial CPU approaches, or even if some modifications to use some degree of synchronization and shared memory management may allow continued increasing of performance.

Table 2: Execution Times and Speedup

Problem Size for Serial and Parallel - GPU Threads for Parallel Only		**Program Execution Time (Seconds)**		**Speedup Relative To Serial**	
Size	**GPU Threads**	**Serial**	**Parallel**	**Serial**	**Parallel**
1,024	1,027	0.000018	0.000293	1	0.061
2,048	2,051	0.000037	0.000294	1	0.126
4,096	4,099	0.000073	0.000335	1	0.218
8,192	8,195	0.000145	0.000290	1	0.500
16,384	16,387	0.000291	0.000312	1	0.933
32,768	32,771	0.000581	0.000422	1	1.377
65,536	65,539	0.001162	0.000533	1	2.180

7 Conclusions

The average of the trials suggests that a parallel GPU based approach can but won't always speed up signal convolution. This may be because the highly independent nature of computing each output matrix value means that the process can potentially benefit highly from the more cores that can be utilized for a larger problem. In contrast, for smaller problems it seems to be the case that the overhead required both to launch the threads, as well as in some cases the additional CUDA overhead to simply first launch a CUDA function at program start combine to make a traditional serial CPU based approach not only suitable but faster to solve these kinds of problems compared to just a fairly straightforward parallel implementation of 1D convolution. Existing libraries exist which employ more advanced techniques to increase performance even further. One potential area worth examining would be whether it is the case that these techniques also only extend to improving performance to scale better for larger datasets at the cost of creating further overhead that keeps performance behind a traditional sequential CPU-based approaches.

References

[1] GeForce GTX 1080 Ti | Specifications | GeForce. `https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/`.

[2] Maverick2 user guide - tacc user portal. `https://portal.tacc.utexas.edu/user-guides/maverick2`.

[3] Stampede2 user guide - TACC user portal. https://portal.tacc.utexas.edu/user-guides/stampede2.

[4] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[5] S. W. Smith. The scientist and engineer's guide to digital signal processing. 1999.

Developing Incident Response-Focused Cybersecurity Undergraduate Curricula *

Junghwan "John" Rhee, Myungah Park, Fei Zuo, Shuai Zhang, Gang Qian, Goutam Mylavarapu, Hong Sung, Thomas Turner
Department of Computer Science
University of Central Oklahoma
Edmond, OK 73034
{jrhee2,mpark5,fzuo,szhang10,gqian,smylavarapu,hsung,trturner}@uco.edu

Abstract

Increasing cybersecurity incidents call for a competitive cybersecurity workforce more than ever. We create new comprehensive cybersecurity university curricula in a metro undergraduate-focused regional university. Our program is distinguished from other programs by specializing in incident response, which addresses the analysis of attack trails and damages followed by infrastructure hardening including software, systems, and policies to prevent future attacks. This special theme requires practical strength such as certifiable deep knowledge, hands-on skills, and research-involved cybersecurity activities. We present the design and implementation of the cybersecurity curricula in our institution.

1 Introduction

Increasing cybersecurity incidents and malware hits as seen in the Colonial pipeline incident [13] have been issues with real impact on our society. Reputable sources such as Cyberseek [6] and ISC^2 [7] from industry leaders estimate 4.7 million cybersecurity experts are lacking. Cybersecurity is becoming increasingly important in computer science education as an essential skill because a poor design without careful consideration of software vulnerabilities and

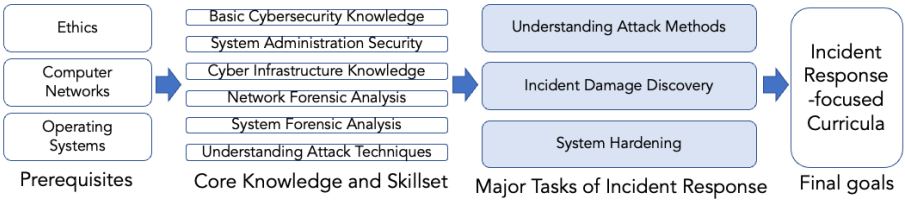


Figure 1: Mapping goals with major tasks, core knowledge, and prerequisites.

attack surfaces can invite adversaries and incidents in no time in the current connected world. This trend is evidenced by multiple movements of integration of cybersecurity into the design and development of software such as Microsoft's Secure Software Development Life Cycle (SDL) [10]. Recent announcements promoting memory-safe languages as a major transition by Microsoft and NSA show cybersecurity is an inseparable subject in computer science education and affects the major decision in computing.

As cybersecurity has been a major social issue with great demand in the workforce, our department created a new cybersecurity program specialized in incident response, which is an essential subject for cybersecurity analysts who are the experts to evaluate and improve each organization's cybersecurity status. This paper proposes a new design of cybersecurity education curricula with the following contributions:

- We introduce a design of an undergraduate cybersecurity program specialized in incident response.
- We present a case study of the design and implementation of this program as a real instance of an incident response-focused cybersecurity program in a regional university.

We present the design of our curricula in Section 2. Related work is discussed in Section 3. We discuss our plan for the next step in Section 4. Our paper is concluded in Section 5.

2 Design of Incident Response-focused Curricula

We set multiple objective requirements for the design of our cybersecurity program to make it distinguished from others.

2.1 Design Requirements

- **R1: Specialized education plan for unique competitiveness:** First, we wanted to make our program unique and distinguished with a unique value in incident response. This requirement can be fulfilled by understanding the necessary skill sets and corresponding course design. Figure 1 presents the mapping that we determined. Core skillsets to conduct incident response include an understanding of cybersecurity attack methods, assessment of attack damages, and system hardening methods to prevent future attacks. We determined the required core skills and knowledge to perform these tasks and support prerequisites.

- **R2: Multiple cybersecurity education targets:** Second, our computer science department has students with multiple different degree interests such as B.S. of Computer Science, B.S. of Software Engineering, B.S. of Data Science, and M.S. of Computer Science. Cybersecurity curricula should be positioned to be beneficial to students who need cybersecurity with different degree goals.

- **R3: Quality hands-on skills:** Cybersecurity employees require strong knowledge and hands-on skills which help them to become effective employees in their tasks investigating incidents and securing the infrastructure which require strong underlying system knowledge.

- **R4: Credentials to strengthen job applications:** The current competitive job market often calls for cybersecurity credentials in job applications. For fresh graduates who start the first position in their job career, this is an important aid to assist their landing positions.

- **R5: Research experience:** Participating in research provides an opportunity to learn the latest technology, challenges, and solutions. Tackling and solving unsolved problems could nurture students as problem solvers. Also for students who seek research career paths, research experience will become an important record to help their applications.

2.2 Design Foci of the Proposed Program

With these requirements in mind, we designed our curricula to satisfy each of the requirements.

- **D1: Focus on incident response:** Our department has multiple faculty members working on cybersecurity. Based on their research and teaching background, we designed and implemented cybersecurity curricula specializing in incident response, as shown in Figure 2. The required

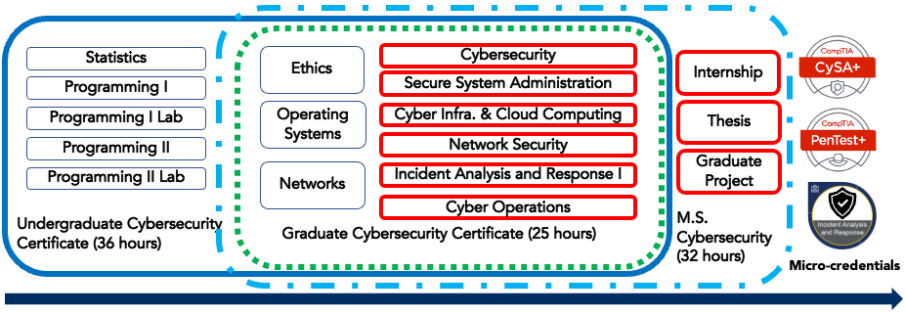


Figure 2: Curricula of proposed courses, degrees, and certificates.

knowledge and skillset shown in Figure 1 are fulfilled by six cybersecurity core courses and three supporting courses. We created an undergraduate certificate that provides comprehensive cybersecurity skillsets with these courses. We are also creating graduate-level programs for students who seek advanced study.

- **D2: Multiple student targets:** We set two types of student targets for our education program: The first target is the frontline of cybersecurity defense. The students of this group intend to work in the cybersecurity positions such as cyber analysts, penetration testers, and cyber operators. The second education target is the computer science majors with cybersecurity awareness and competitiveness. For the second group, our program is positioned to provide strong add-on cybersecurity skills for their main career goals to make them more competitive.

- **D3: Combination of hands-on skills and deep knowledge:** Hands-on assignments and projects provide an opportunity to deeply understand the learning materials and apply what they have learned into practice. We train our students to be competitive in skills as well as knowledge so that their capabilities are practical and influential in their positions. For instance, the Secure System Administration course teaches essential utilities, command line interfaces, and scripting (bash, PowerShell) based on Windows and Linux virtual machines. The Incident Analysis and Response I course covers the collection of operating system data and their investigation. Students analyze the data using various UNIX tools and scripting to understand attack details and make an incident report. The Cyber Operation course teaches penetration testing to discover the weaknesses of systems and software using a comprehensive set of tools. Also, students participate in multiple catch-the-flag (CTF) events with

the instructor and experience vulnerability discovery in the field.

- **D4: Cybersecurity credentials:** Our department became a partner of CompTIA [5], which is a well-known cybersecurity organization in the industry and government. Our curriculum is closely aligned with its certificates. Besides, in several courses, we introduced micro-credentials that can be validated regarding the details of skills obtained. We also are preparing applications for accreditation such as NSA National Centers for Academic Excellence in Cybersecurity [12].

- **D5: Research-involved education program:** Our department has multiple faculty members who are active in cybersecurity research. We offered multiple research assistant opportunities almost every semester providing research experience. Their work is submitted to research publications and research events such as Oklahoma Research Day.

2.3 Degrees and Certificates

We created multiple cybersecurity programs as follows.

- **Undergraduate cybersecurity certificate**: Undergraduate level cybersecurity certificate is offered to the students who complete a list of core subjects in cybersecurity and supporting courses. This certificate will provide a cybersecurity credential for undergraduate students (D2).

- **Micro-credentials:** Multiple courses offer micro-credentials for their successful completion. These are issued using a third-party service called "Credly" so that a digital certificate can show the details regarding the earned skill (D4) by following the URL links assigned to the issued micro-credentials.

- **M.S. in Cybersecurity:** We submitted an application for a new Master's degree in Cybersecurity to provide advanced education for students who seek the next level. Currently, it is in the review process.

- **Graduate cybersecurity certificate:** We also submitted an application for a graduate-level cybersecurity certificate to provide a cybersecurity credential without the full commitment of the master's degree due to work and other reasons. This is also under review.

2.4 Cybersecurity Courses

We offer multiple courses specialized in cybersecurity in our department.

- **Cybersecurity:** This course introduces the foundations of cybersecurity. The course topics include cyber threats, security principles and goals, security policies and mechanisms, access control, cryptography, operating system and software security, and legal and ethical issues in cybersecurity.

- **Network Security:** This course introduces the principles of network security, which covers network security policy, packet filtering, firewalls, port scanning, intrusion detection and prevention systems, virtual private networks, DNS security, host hardening, and network incident response.

- **Secure System Administration:** This course introduces the essential knowledge and skillsets for system administration. Students will learn hands-on system administration techniques including scripting, shells, editors, utilities, policies, regulations, and risk management (D3).

- **Incident Analysis and Response I:** This course introduces the knowledge and skillsets for incident response, system analysis, and security controls. Students will learn hands-on techniques to investigate the symptoms of attacks and perform a comprehensive analysis to discover the details of attack damages, recover the systems, and protect them from future attacks. This course covers the materials for the CompTIA CySA+ certificate (D1, D3, D4).

- **Cyber Operations:** This course introduces techniques and hands-on experiences in penetration testing, software security, and cyber operations. Students will learn vulnerable/secure code patterns, vulnerability types, and security frameworks for vulnerability testing. This course covers the materials for the CompTIA PenTest+ certificate (D3, D4).

- **Cyber Infrastructure and Cloud Computing:** This course introduces the technologies of cloud computing, the stack of cloud service models, and the design of cloud-based applications. Hands-on studies over prevailing cloud services are provided to illustrate the common components, interfaces and practices for resource management and application development at scale.

2.5 Extra Activities

In addition to the presented curricula, we created several supporting activities and groups to enhance cybersecurity education.

- **Cybersecurity Center:** We established a cybersecurity center, which serves as a hub for cybersecurity activities at our institution (D5).

- **Cybersecurity Research Groups:** Our faculty members run research labs to lead cybersecurity research and educational projects with undergraduate and graduate students (D5).

- **Cybersecurity Community Activities:** We are participating in multiple cybersecurity community activities such as Catch the Flags (CTF) events. We are also collaborating with industry and government organizations forming an advisory board.

3 Related Work

There are multiple papers regarding the design of cybersecurity curricula and the survey of cybersecurity programs related to this paper.

Design of Cybersecurity Curricula: Santos et al. [20] present the reflections regarding the curricula contents to be considered to design a graduate-level curriculum in cybersecurity. In [21], Schneider et al. argue the issues about what should be taught and which are not well addressed by many of the university cybersecurity faculty. Rowe et al. [17] discussed the role of cybersecurity in an IT education context and argue why IT programs should champion this topic. Rashid et al. [16] present the Cyber Security Body of Knowledge (CyBOK) project that classifies the foundational and generally recognized knowledge on cybersecurity. Blaken-Webb et al. [2] describe the rationale for and implementation of an experimental graduate-level cybersecurity ethics course curriculum recently piloted at their school. Blair et al. [1] present a vision and curricula for multi-disciplinary cybersecurity teams which are made up of experts of diverse abilities and expertise. Yuan et al. [24] provide a detailed account of designing and developing a hands-on cybersecurity project, which consists of penetration testing, defending an enterprise-level network system, and performing a comprehensive IT audit. The authors of [8] propose a course of study in cybersecurity designed to target homeland security students. The curriculum promotes the intellectual strengths of students in this discipline and that is consistent with the broad suite of professional needs. Saharinen et al. [19] describe a model for designing a degree program in cybersecurity. The authors establish the guiding frameworks and requirements within the European Union for a degree program. Sharevski et al. [22] developed an interdisciplinary course for learning in the fields of cybersecurity and interaction design. The inaugural course teaches students secure user interface design. Yue et al. [25] present a case study of an ABET-accredited cybersecurity program. Luallen et al. [9] discuss the development of a course and laboratory environment regarding an undergraduate and graduate-level critical infrastructure and control systems cybersecurity curriculum.

While there have been multiple approaches proposed for cybersecurity curricula, our proposal is differentiated from them because it is a cybersecurity education program with a specialty in incident response education for undergraduate students.

Survey of Cybersecurity Education Programs: There are prior surveys on cybersecurity education programs. Multiple researchers [3, 4, 11, 15, 18] present an overview and comparison of existing curriculum design approaches for cybersecurity education as a survey and review. Yamin et al. [23] aim to identify commonalities in skillset requirements for multiple cybersecurity roles like penetration tester, security operation center analysts, digital forensic and incident responders, and information security managers. They determined five main domains which are all included in our curriculum.

Cybersecurity Education Framework: NIST's NICE framework [14] aims to create an operational, sustainable, and continually improving program for cybersecurity awareness, education, training, and workforce development that advances the US's long-term cybersecurity posture. This framework provides a foundation for most cybersecurity education programs.

4 Discussion

This paper presents the design and implementation of our cybersecurity program. After its long-term operation for multiple years, we plan to publish the outcome of our program and reflection on the result including further enhancement, modifications, student career, and lessons from the operation as future work.

5 Conclusion

We propose cybersecurity curricula focused on the incident response which is an essential task for cybersecurity workforce to address security incidents and secure infrastructure. We designed the curricula by determining core knowledge and skillset for the major tasks of incident response and then mapping them to six core cybersecurity courses. Our institution implemented this design and offers an undergraduate cybersecurity certificate. In addition, currently, we are in the process of creating a graduate-level master's degree and cybersecurity certificate as advanced steps. We hope our design and experience could be helpful to the institutions that plan a new cybersecurity program.

References

[1] Jean R.S. Blair, Andrew O. Hall, and Edward Sobiesk. Educating future multidisciplinary cybersecurity teams. *Computer*, 52(3):58–66, 2019.

[2] Jane Blanken-Webb, Imani Palmer, Sarah-Elizabeth Deshaies, Nicholas C Burbules, Roy H Campbell, and Masooda Bashir. A case study-based cybersecurity ethics curriculum. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*, 2018.

[3] Krzysztof Cabaj, Dulce Domingos, Zbigniew Kotulski, and Ana Respício. Cybersecurity education: Evolution of the discipline and analysis of master programs. *Computers & Security*, 75:24–35, 2018.

[4] Nabin Chowdhury and Vasileios Gkioulos. Cyber security training for critical infrastructure protection: A literature review. *Computer Science Review*, 40:100361, 2021.

[5] CompTIA. Computing technology industry association (CompTIA). `https://www.comptia.org/`.

[6] CompTIA. Cyberseek. `https://www.cyberseek.org/`.

[7] ISC^2. 2022 cybersecurity workforce study. `https://www.isc2.org/Research/Workforce-Study`.

[8] Gary C. Kessler and James D. Ramsay. A proposed curriculum in cybersecurity education targeting homeland security students. In *2014 47th Hawaii International Conference on System Sciences*, 2014.

[9] Matthew E. Luallen and Jean-Philippe Labruyere. Developing a critical infrastructure and control systems cybersecurity curriculum. In *2013 46th Hawaii International Conference on System Sciences*, 2013.

[10] Microsoft. Microsoft security development life cycle. `https://www.microsoft.com/en-us/securityengineering/sdl`.

[11] Djedjiga Mouheb, Sohail Abbas, and Madjid Merabti. *Cybersecurity Curriculum Design: A Survey*, pages 93–107. 04 2019.

[12] NSA. National centers of academic excellence in cybersecurity. `https://www.nsa.gov/Academics/Centers-of-Academic-Excellence/`.

[13] Energy Security Office of Cybersecurity and Emergency Response. Colonial pipeline cyber incident. `https://www.energy.gov/ceser/colonial-pipeline-cyber-incident`.

[14] Celia Paulsen, Ernest McDuffie, William Newhouse, and Patricia Toth. NICE: Creating a cybersecurity workforce and aware public. *IEEE Security & Privacy*, 10(3):76–79, 2012.

[15] Denny Pencheva, Joseph Hallett, and Awais Rashid. Bringing cyber to school: Integrating cybersecurity into secondary school education. *IEEE Security & Privacy*, 18(2):68–74, 2020.

[16] Awais Rashid, George Danezis, Howard Chivers, Emil Lupu, Andrew Martin, Makayla Lewis, and Claudia Peersman. Scoping the cyber security body of knowledge. *IEEE Security & Privacy*, 16(3):96–102, 2018.

[17] Dale C. Rowe, Barry M. Lunt, and Joseph J. Ekstrom. The role of cybersecurity in information technology education. In *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, 2011.

[18] Rodrigo Ruiz. A study of the uk undergraduate computer science curriculum: A vision of cybersecurity. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, 2019.

[19] Karo Saharinen, Mika Karjalainen, and Tero Kokkonen. A design model for a degree programme in cyber security. In *Proceedings of the ICETC 2019*.

[20] Henrique Santos, Teresa Pereira, and Isabel Mendes. Challenges and reflections in designing cyber security curriculum. In *2017 IEEE World Engineering Education Conference (EDUNINE)*, pages 47–51, 2017.

[21] Fred B. Schneider. Cybersecurity education in universities. *IEEE Security & Privacy*, 11(4):3–4, 2013.

[22] Filipo Sharevski, Adam Trowbridge, and Jessica Westbrook. Novel approach for cybersecurity workforce development: A course in secure design. In *2018 IEEE Integrated STEM Education Conference (ISEC)*, 2018.

[23] Muhammad Mudassar Yamin and Basel Katt. Cyber security skill set analysis for common curricula development. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, New York, NY, USA, 2019. ACM.

[24] Dongqing Yuan. Design and develop hands on cyber-security curriculum and laboratory. In *2017 Computing Conference*, pages 1176–1179, 2017.

[25] Xiaodong Yue, Belinda Copus, and Hyungbae Park. How to secure ABET accreditation for a cybersecurity program: A case study. *J. Comput. Sci. Coll.*, 2022.

Hands-On Lab Development for Policy Violations in Voice Personal Assistants*

Alejandra Enriquez Sanchez, Oludare Ogunbowale
Olayinka Adetola, Na Li
Computer Science Department
Prairie View A&M University
Prairie View, TX 77446
{aenriquezsanchez, nali}@pvamu.edu

Abstract

This paper discusses the design and implementation of a hands-on lab for undergraduate students to learn Voice Personal Assistants(VPA), their policies, and possible policy violations in existing Amazon Alexa skills and Google Assistant actions. A pilot lab session and a survey were conducted among 14 undergraduate students enrolled in the class, Introduction to Information Security, at Prairie View A&M University in the fall of 2022. Students' feedback was very positive, demonstrating the effectiveness of the lab for them to learn relevant knowledge and skills.

1 Introduction

The rise of Voice Personal Assistants (VPA) in the cloud-based artificial intelligence and IoT space has been propelled by market leaders such as Amazon's Alexa, Apple's Siri, Google Assistant, and Microsoft's Cortana. VPAs allows users to communicate through speech using customized software. However, the increased usage of VPAs in households raises concerns about users' safety and privacy disclosure. Although Amazon Alexa and Google Assistant have policies in place to protect their users, research has shown that only 10% of users are definitely aware of what data can be collected[7]. It also shows a misplaced

trust in Amazon and Google's certification process to check and detect any policy violations before the publishing of the action or skill.

This paper focuses on Amazon Alexa skills and Google Assistant actions, two of the most widely known VPA applications. The Amazon Alexa and Google Assistant platforms [3][9] allow third-party developers to publish their VPA applications (skills or actions). Both of the platforms perform a certification process to assess the submitted skills or actions and determine whether they are appropriate for publishing. Despite the existence of this process, some published skills/actions still violate the policies set by Amazon and Google. Considering the popularity of VPAs, especially among younger generations, we were motivated to educate students on VPA policies and possible violations by developing hands-on lab activities. Particularly, we intended to (1) introduce VPA policies to students; (2) raise students' awareness of possible policy violations in using VPAs; (3) teach them how to properly test VPA applications; and (4) empower them to discover policy violations on their own.

The rest of the paper is organized as follows: Section 2 briefly discusses literature related to policy violations in VPA and the skill certification process. Section 3 presents the development of the lab. Section 4 discusses the survey evaluation. Finally, a conclusion is made in Section 5.

2 Related Work

2.1 Information Leakage From Third-Party VPAs

Research exposes a lack of security measures to prevent sensitive information leakage through third-party voice applications. Bispham et al. interacted with Google Actions and Alexa skills via a chat box and confirmed the leakage of sensitive information[5]. They believe it remains a challenge to thoroughly prevent such leakage through the conversation interface of third-party voice applications due to the current architecture of voice assistance. They suggested redesigning the voice assistant architecture to prevent future security risks.

Sabir et al. also discovered a significant gap in users' understanding of how Alexa selects and auto-enables skills[13]. This study exposes the lack of proper auditory interventions which are necessary to minimize VPA applications' security and privacy risks.

2.2 Trustworthiness of Skill Certification in VPAs

Amazon and Google require skills and action developers to pass a certification to publish their applications. However, findings demonstrate the policy requirements are not strongly imposed despite claims. Cheng et al. designed and submitted 234 Alexa skills that deliberately violated 55 content and privacy

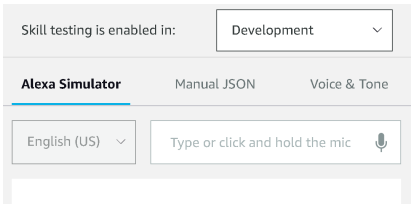


Figure 1: Alexa console

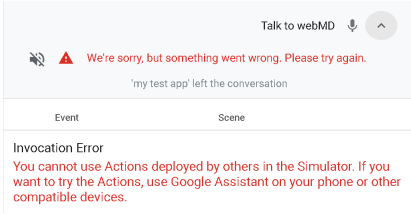


Figure 2: Error display when testing existing action

policies defined, all of which passed the certification process[7]. In comparison, they also submitted 381 policies violating Google actions, out of which 39% passed the certification process. They manually tested 755 Alexa skills in the Kid's category and identified 31 problematic skills with policy violations and 34 broken skills. They tested all 114 actions in the same category and only found one with a policy violation. Additionally, Liao et al. investigated and analyzed the overall effectiveness of privacy policies provided by developers through an NLP-based analysis approach [11].

They systematically measured the effectiveness of privacy policies provided by voice-app developers in both Alexa and Google stores to understand the quality and usability issues of privacy policies provided by developers. They also conducted a user study to understand users' perspectives on VPA's privacy policies. They found that there exist a substantial number of problematic privacy policies. Although research has been conducted on VPA policies, education on this topic for younger undergraduate students has not been sufficiently emphasized.

3 Lab Development

In this section, we will detail how we developed the lab activities, including the investigation of the environmental setup, researching VPA policies in the Kids and Healthcare category, and testing several VPA skills and actions. Additionally, we will briefly introduce the structure of the lab manual we designed for students.

3.1 VPA Testing Platforms

VPA services like Amazon Alexa and Google Assistant offer a development console for developers to create, test, manage and publish skills and actions. Through the console, developers can simulate the interaction between a skill

or action and an end user. The simulator allows both voice and text input for this purpose [2].

The console interface for Amazon Alexa is displayed in Figure 1. Note that there is an error if one tries to test an existing action developed by others, as displayed in Figure 2. This is because Google does not allow a developer to test any actions through the console which do not belong to the developer himself. Therefore, we tested some existing actions with the Assistant mobile application available in Google Play store and Apple store, instead of using the console.

3.2 VPA Policy Investigation

We investigated the policies in general and their different categories created by both Amazon and Google. Then we decided to focus on two of the most critical categories, *Kids* and *Healthcare*.

The Amazon Alexa platform defines three specific policies for skills in the Kids category as listed in Table 1. The Google Assistant platform specifies the first and third policies above as well, but it does not explicitly forbid actions from directing its users to external websites[4][10]. In the Healthcare category, both Amazon Alexa and Google Assistant platforms defined the two policies listed in Table 1. Amazon does not allow skills to collect information about a user's physical or mental health. Google requires that actions cannot collect information that could be considered protected health information (PHI) under the Health Insurance Portability and Accountability Act (HIPAA). For both categories, Amazon Alexa requires skills in data collection to provide a privacy policy/notice, whereas Google Assistant requires every action to have a privacy policy/notice.

Table 1: Policy Definitions

Category	Policy Requirements by VPA Platforms
Kids	It cannot collect any personal information from end users.
	It cannot promote any products, content, or services or direct end users to engage with content outside of Alexa.
	It cannot include content that is not suitable for all ages.
Health	It cannot collect information relating to any individual's physical or mental health condition.
	If a skill provides health-related information, it has to include a disclaimer that it is not substitute for medical advice.

3.3 VPA Testing

We tested several existing skills and actions manually. Specifically, we attempted to violate each existing policy through every possible interaction path. For Amazon Alexa, 20 skills were tested in the Kid's category, out of which three were found with violations. Additionally, 20 skills were tested in the

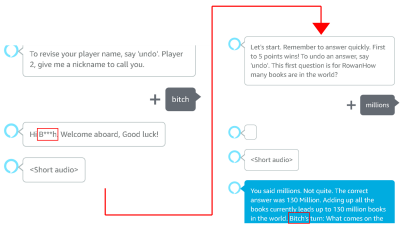


Figure 3: Quiz Yay skill

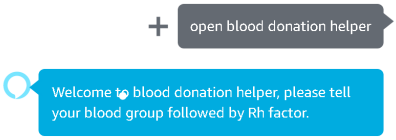


Figure 4: Blood donation helper skill

Healthcare category, out of which two were found in violation. For Google Assistant, ten actions were tested in the Kids category, out of which the authors found no violation. Another ten actions were tested in the Healthcare category, out of which two actions were found with violations. Due to the space limit, we selected two skills and two actions with policy violations for illustration in the following.

3.3.1 Skill I

The first skill selected is "Quiz Yay", which belongs to the Kid's category and is a simple game to entertain kids [1]. The game starts by asking how many players will be playing, followed by requesting a name from each player. The players can use any name, including inappropriate words such as curse words. While the game attempts to censor the word upon selection, the skill utilizes the uncensored word during that player's turn, as circled in Figure 3. This is a violation against the second policy in the Kids category in Table 1.

3.3.2 Skill II

The second skill selected is "Blood donation helper" in the Healthcare category [8]. This skill helps the blood donor choose the people who have the correct receiving blood group. However, its initial response, as pictured in Figure 4, violates the first policy in the Healthcare category in Table 1, as a person's blood type is considered personal data and should not be collected by the skill. Although knowing the correct blood type is important for safe transfusion, the skill can simply demonstrate the correct matching between blood types and receiving groups without inquiring about the user's blood type.

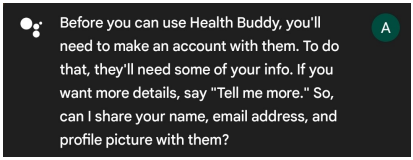


Figure 5: Health Buddy action

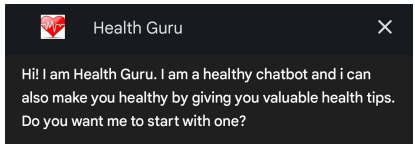


Figure 6: Health Guru action

3.3.3 Action I

The first action selected is “Health Buddy" in the Healthcare category [6]. This action helps users track calories of the food they consume daily. Upon interaction, it asks a user for his name, email, and profile picture before he can access the action, as pictured in Figure 5. This violates Google’s health policy which does not allow actions to provide, collect, or store personal medical information, including data that could be considered data concerning health under the General Data Protection Regulation.

3.3.4 Action II

The second action selected is “Health Guru" in the Healthcare category pictured [12]. This action claims to help users lead a healthier lifestyle if they follow their tips and fact. Although the action displays a disclaimer in the description that states, “this app is not a substitute for medical advice...", it fails to make a disclaimer upon first interaction with the user as seen in Figure 6. Google’s policy requires actions that provide health information, such as non-personalized information about symptoms, treatments, or medications include a disclaimer at the beginning of the user’s first interaction with the action and in the Directory description. Therefore, it causes a policy violation.

3.4 Lab Manual Design

The lab manual includes instructions on how to set up the environment to test skills and actions and illustrates how to break the policies of the selected skills and actions aforementioned. Particularly, students can read the information about those VPA applications from their official sites and they are able to interact with them to replicate the scenarios of policy violations. The manual also explains why they break the policies. Ultimately, the task encourages students to explore more about detecting policy violations of VPAs by testing more skills and actions.

4 Survey Evaluation

To evaluate the effectiveness of our lab, we piloted a lab session in a lower-level Computer Science elective class for undergraduate students, *Introduction to Information Security*, at Prairie View A&M University in the end of Fall 2022. A total of 14 students participated in the lab session. During the lab session, the students were given a brief background lecture explaining what VPAs are, their policies, the importance of VPA policies, the impact of policy violations, and some examples of policy violations. Then students were walked through the activities in the lab manual. We also conducted pre and post-surveys for the lab. The survey questions were designated to measure the students' gain in awareness, interest, and understanding of the topic, as listed in Table 2. The survey questions were divided into two categories, questions in both pre and post-surveys, and questions only in the post-survey. All questions use a rating scale of one to five, with five being the most positive.

Table 2: Survey questions for evaluating the labware

#	Survey Question	Type
1	Consider your level of awareness about Policies of VPA services (P-VPA)	Pre and Post
2	Consider your level of awareness about possible Privacy Disclosure of VPA applications (PD-VPA)	Pre and Post
3	Consider your level of awareness about Policy Violations of VPA applications (PV-VPA)	Pre and Post
4	Consider your level of interest in Testing VPA applications	Pre and Post
5	Consider your level of interest in Detecting Policy Violations of VPA applications	Pre and Post
6	Consider your level of interest in Developing VPA applications with Policy Compliance	Pre and Post
7	Indicate the extent of your gains in understanding Voice Personal Assistant (VPA) services	Post
8	Indicate the extent of your gains in understanding Policies in VPA applications (Kids and Healthcare categories)	Post
9	Indicate the extent of your gains in understanding Policy Violations in VPA applications	Post
10	This lab helped me understand how to test VPA applications.	Post
11	The lab helped me understand what policies should be followed for developing a VPA application.	Post
12	The lab helped me understand how to detect policy violations of VPA applications.	Post
13	I would like this lab to be taught in a computer security course.	Post

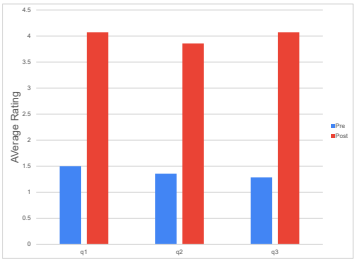


Figure 7: Student's level of aware- ness

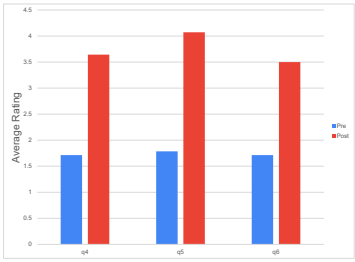


Figure 8: Student's level of interest

4.1 Category I

The questions in this category are focused on assessing students' level of awareness and interest in different relevant concepts.

For each question in Questions 1 to 6, the average rating of the students' responses was calculated for the pre and post-surveys. Questions 1 to 3 focused on students' awareness of the VPA policies, possible privacy disclosure, and policy violations of VPA applications. Questions 4 to 6 focused on students' interest in testing and detecting policy violations of VPA applications and developing VPA applications with policy compliance. Figures 7 and 8 demonstrate a significant increase in student's awareness and interest levels after the lab session with the average rating being above 4 or close to 4.

4.2 Category II

Questions in this category were designed to gather students' feedback on their understanding gained after the lab session and on the effectiveness of the lab. The final question sought to gauge the interest in including this lab and topic in a computer security course.

Questions 7 to 9 sought to gauge the extent of the students' gain in understanding VPA, policies in VPA applications, and policy violations. As seen in Figure 9, the students indicated a significant gain in understanding of those concepts. Questions 10 to 12 sought to assess whether the lab helped the students understand how to test VPA applications, what policies should be followed for developing a VPA application, and how to detect policy violations. Figure 10 indicates a strong agreement that the lab helped the students learn the corresponding knowledge and skills.

The last question in the post-survey can help us understand if this topic

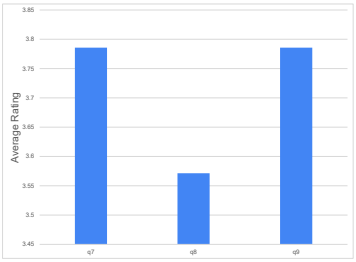


Figure 9: Student's gain in understanding

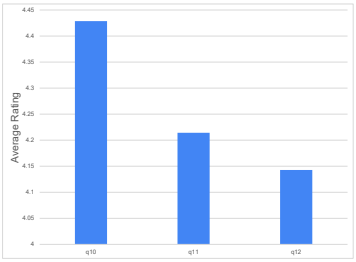


Figure 10: This lab helped students with understanding the topic

interests the students and whether they are willing to learn through lectures and hands-on experience like the lab presented. Most students agreed that they would like this lab to be taught in a computer security course (6 strongly agree, 6 agree, and 2 neither agree nor disagree).

5 Conclusion

This paper discusses the development of a lab intended to make students aware of the possible privacy violations in using VPAs, expose them to VPA policies, teach them how to test VPA applications, and help them discover privacy violations. We explored Amazon Alexa and Google Assistant policies and tested some of their skills and actions. We designed the lab manual and piloted the lab in a class with a survey distributed to the students. The survey feedback is very positive, indicating the effectiveness of the lab activities we designed for students to learn all relevant knowledge and skills.

Acknowledgment

This project is supported in part by the National Science Foundation (NSF) under grant DUE-1712496. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of NSF. Additionally, we thank Mr. Okechukwu Ogwo-Ude for his assistance in researching skills and running the lab session.

References

[1] Aarna. Quiz yay. https://www.amazon.com/Aarna-Quiz-yay/dp/B09WVLY66H/.

[2] Amazon. Alexa developer console. https://developer.amazon.com/alexa/console/ask.

[3] Amazon. Alexa skills. https://www.amazon.com/alexa-skills/b/?ie=UTF8&node=13727921011.

[4] Amazon. Policy testing. https://developer.amazon.com/en-US/docs/alexa/custom-skills/policy-testing-for-an-alexa-skill.html.

[5] Mary Bispham, Clara Zard, Suliman Sattar, Xavier Ferrer-Aran, Guillermo Suarez-Tangil, and Jose Such. Leakage of sensitive information to third-party voice applications. In *Proceedings of the 4th Conference on Conversational User Interfaces*, pages 1–4, 2022.

[6] Bonitoz.Inc. Health buddy. https://assistant.google.com/services/a/uid/00000075a4065a36.

[7] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. Dangerous skills got certified: Measuring the trustworthiness of skill certification in voice personal assistant platforms. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1699–1716, 2020.

[8] Leslie Correa. Blood donation helper. https://www.amazon.com/Leslie-Correa-Blood-donation-helper/dp/B07N626993.

[9] Google. Assistant actions. https://assistant.google.com/explore.

[10] Google. Policies for actions on google. https://developers.google.com/assistant/console/policies/general-policies.

[11] Song Liao, Christin Wilson, Long Cheng, Hongxin Hu, and Huixing Deng. Measuring the effectiveness of privacy policies for voice assistant applications. In *Annual Computer Security Applications Conference*, pages 856–869, 2020.

[12] Sandeep Kumar Rana. Health guru. https://assistant.google.com/services/a/uid/000000114f6751c7.

[13] Aafaq Sabir, Evan Lafontaine, and Anupam Das. Hey Alexa, who am I talking to?: Analyzing users' perception and awareness regarding third-party Alexa skills. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2022.

Programming Many-Core Architectures (GPUs) Using CUDA*

Conference Tutorial

Eduardo Colmenares
Computer Science
Midwestern State University
Wichita Falls, Texas, 76308
eduardo.colmenares@msutexas.edu

Many-core devices also known as Graphical Processing Units (GPUs) have dominated the floating point race since 2013 and their benefits have been fueling the current artificial intelligence (AI), machine learning (ML), deep learning (DL), and data science (DS) revolution. Many of these fields benefit from the use of specialized frameworks, such as Tensor Flow, Keras, etc. Although extremely helpful for AI, ML, DL and DS related purposes, the use of these frameworks does not provide the user any knowledge about how to harness the potential of the underlying hardware for different applications or different fields to those mentioned above. This tutorial intends to provide interested students and faculty a basic, but strong hands-on programming foundation regarding good practices and potential capabilities of modern GPUs. The tutorial focuses on GPU programming and not AI Frameworks.

Designing Learning Outcomes and Competencies using Bloom's for Computing*

Conference Tutorial

Markus Geissler, Koudjo Koumadi, Pam Schmelz
Christian Servin, Cara Tang and Cindy Tucker
Association of Computing Machinery
Committee for Computing Education in Community Colleges

In this tutorial, participants will be introduced to Bloom's for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines, a project of the ACM CCECC (Committee for Computing Education in Community Colleges) [1, 2, 4, 3]. Due for final publication by the end of 2022, the Bloom's for Computing report offers a total 57 additional verbs across all six levels of Bloom's cognitive domain – Remembering, Understanding, Applying, Analyzing, Evaluating, Creating. The enhanced verb list is intended to support crafting more appropriate and less awkward learning outcomes and competencies that express the knowledge, skills, and dispositions required in computing disciplines. The Bloom's for Computing verb list and report is not just for use in future ACM curriculum guideline reports, but is primarily for educators in computing disciplines who find themselves needing to craft learning outcomes or competencies – whether for programs, courses, or individual modules; whether two-year, four-year, graduate, or K-12 level; whether faculty, instructional designers, or program coordinators.

The presentation and activities in the proposed tutorial session are outlined below:

1. Introductions – tutorial facilitators & participants
2. Refresher on Bloom's Revised Taxonomy, its 6 cognitive levels, and common verbs lists

3. Interactive discussion on how faculty approach writing learning outcomes and some of the challenges encountered

4. Bloom's for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines

   (a) Introduce the project & the verbs

   (b) Examples of learning outcomes using the Bloom's for Computing verbs

   (c) Areas where the Bloom's for Computing verbs come in particularly handy

5. Activity where participants write or modify learning outcomes for courses they teach

6. Share out learning outcomes and thoughts on how the enhanced verbs might be used

7. Wrap up

Participants will be given a handout to take home with the complete list of verbs for each cognitive level.
This tutorial is relevant for anyone involved in writing, revising, or updating learning outcomes for programs, courses, or instructional units in computing disciplines such as Computer Science, Information Technology, and Cybersecurity.

Biography

Christian Servin is an Associate Professor of Computer Science at El Paso Community College; he has more than 15 years of experience teaching computing courses, developing best practices in computing education, and establishing academic and research partnerships between community colleges with ISDs and four-year colleges. Currently, he serves as a vice chair for the ACM Standing Committee for Computing Education in Community Colleges (CCECC), where he assists in the development and updates curricular guidelines in computing for 2-Year colleges and computing education projects. He serves as a member of several committees for The Texas Higher Education Coordinating Board representing Two-Year Colleges. He has served in various curricular guidelines recommendations such as the ACM Data Science Task Force and the CS2023: ACM/IEEE-CS/AAAI Computer Science Curricula

References

[1] Markus Geissler, Christian Servin, Cara Tang, and Koudjo Koumadi. Operationalizing computing verb enhancements to bloom's revised taxonomy: Leveraging bloom's for computing to create learning outcomes for various computing disciplines. In *Proceedings of the 23rd Annual Conference on Information Technology Education*, SIGITE '22, pages 44–45, New York, NY, USA, 2022. Association for Computing Machinery.

[2] Christian Servin, Cara Tang, Markus Geissler, Melissa Stange, and Cindy Tucker. Enhanced verbs for bloom's taxonomy with focus on computing and technical areas. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, page 1270, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Cara Tang, Markus Geissler, and Christian Servin. Bloom's for computing: Crafting learning outcomes with enhanced verb lists for computing competencies. *J. Comput. Sci. Coll.*, 38(1):114–115, nov 2022.

[4] Cara Tang, Markus Geissler, Christian Servin, and Cindy Tucker. Computing verbs to enhance bloom's revised taxonomy. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2022, page 1026, New York, NY, USA, 2022. Association for Computing Machinery.