The Journal of Computing Sciences in Colleges

Papers of the 28th Annual CCSC Central Plains Conference

April 1st-2nd, 2022
Drury University
Springfield, MO

Baochuan Lu, Editor
Southwest Baptist University

Bin Peng, Associate Editor
Park University

Joseph Kendall-Morwick, Regional Editor
Park University

Volume 37, Number 6 **April 2022**

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

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

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

Karina Assiter, President (2022), (802)387-7112, karinaassiter@landmark.edu.

Chris Healy, Vice President (2022), chris.healy@furman.edu, Computer Science Department, 3300 Poinsett Highway Greenville, SC 29613.

Baochuan Lu, Publications Chair (2024), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Division of Computing & Mathematics, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2023), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

Cathy Bareiss, Membership Secretary (2022), cathy.bareiss@betheluniversity.edu, Department of Mathematical Engineering Sciences, 1001 Bethel Circle, Mishawaka, IN 46545.

Judy Mullins, Central Plains Representative (2023), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, UMKC, Retired.

Michael Flinn, Eastern Representative (2023), mflinn@frostburg.edu, Department of Computer Science Information Technologies, Frostburg State University, 101 Braddock Road, Frostburg, MD 21532.

David R. Naugler, Midsouth Representative(2022), (317) 456-2125, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

Grace Mirsky, Midwest Representative(2023), gmirsky@ben.edu, Mathematical and Computational Sciences, 5700 College Rd. Lisle, IL 60532.

Lawrence D'Antonio, Northeastern Representative (2022), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

Shereen Khoja, Northwestern Representative(2024), shereen@pacificu.edu, Computer Science, 2043 College Way, Forest Grove, OR 97116.

Mohamed Lotfy, Rocky Mountain Representative (2022), Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Tina Johnson, South Central Representative (2024), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308.

Kevin Treu, Southeastern Representative (2024), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.

Bryan Dixon, Southwestern Representative (2023), (530)898-4864, bcdixon@csuchico.edu, Computer

Science Department, California State University, Chico, Chico, CA 95929-0410.

Serving the CCSC: These members are serving in positions as indicated:
Bin Peng, Associate Editor, (816)584-6884, bin.peng@park.edu, Park University - Department of Computer Science and Information Systems, 8700 NW River Park Drive, Parkville, MO 64152.
George Dimitoglou, Comptroller, (301)696-3980, dimitoglou@hood.edu, Dept. of Computer Science, Hood college, 401 Rosemont Ave. Frederick, MD 21701.
Carol Spradling, National Partners Chair, (660)863-9481, carol.spradling@gmail.com, 760 W 46th St, Apt 208, Kansas City, MO 64112.
Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.
Ed Lindoo, Associate Treasurer & UPE Liaison, (303)964-6385, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, 3333 Regis Boulevard, Denver, CO 80221.

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 2022 CCSC Central Plains Conference

Welcome to Springfield, Missouri and Drury University for the Twenty-Eight Annual Consortium for Computing Sciences in Colleges Central Plains Conference. The conference is held in cooperation with the ACM SIGCSE.

Our program features two distinguished speakers, Michael Newton, Software Scientist/Chief Engineer at L3Harris Technologies and Mark Garton, Senior Director of IT Governance and Data Strategy at O'Reilly Auto Parts. The conference has an engaging program featuring paper presentations, tutorials, panels, "Nifty Assignments", Lightning Talks, student poster presentations, student papers, and a student programming contest. The conference starts with a pre-conference workshop on Friday morning. Plenary sessions, paper sessions, panels, and tutorials follow on Friday afternoon and Saturday morning. The program closes with our annual student programming contest on Saturday afternoon.

Many hands are required to organize a conference and I am indebted to many. The conference would not have been possible without the work of an invested conference committee. Holding an in-person conference, after being forced to cancel the conference in 2020 and offering a virtual conference in 2022, would not have been possible without the help of my colleagues, the staff, and the student volunteers at Drury University who have taken care of many of the local details of hosting a conference. Finally, the dedicated work of our reviewers made it possible to select papers using a double-blind process. We accepted 6 of 10 papers submitted for an acceptance rate of 60%.

On behalf of the conference committee I hope you find the conference informative and engaging, meet new colleagues, and take home new ideas to use in your classroom. If you are interested in volunteering for our conference, I encourage you to attend the CCSC:CP Business Meeting Saturday. I look forward to seeing you in 2023 at the University of Central Missouri.

Scott Sigman
Drury University
Conference Chair

2022 CCSC Central Plains Conference Steering Committee

Conference Chair
Scott Sigman ..... Drury University
Conference Co-Chair
Mahmoud Yousef ..... University of Central Missouri
Conference Publicity
Michael P. Rogers ..... University of Wisconsin Oshkosh
Charles Riedesel ..... University of Nebraska-Lincoln
Deepika Jagmohan ..... St. Charles Community College
Farid Nait-Abdesselam ..... University of Missouri Kansas City
Keynote Speakers
Scott Sigman ..... Drury University
Pre-Conference Workshop
Wen Hsin ..... Park University
Michael P. Rogers ..... University of Wisconsin Oshkosh
Judy Mullins ..... Retired
Papers
Mahmoud Yousef ..... University of Central Missouri
Ron McCleary ..... Retired
Panels, Tutorials, Workshops
Bin Peng ..... Park University
Ron McCleary ..... Retired
Nifty Assignments
Mahmoud Yousef ..... University of Central Missouri
Michael P. Rogers ..... University of Wisconsin Oshkosh
Lightning Talks
Diana Linville ..... Northwest Missouri State University
K-12 Outreach, Nifty Assignments & Lightning Talks
Mahmoud Yousef ..... University of Central Missouri
Belinda Copus ..... University of Central Missouri
Shannon McMurtrey ..... Drury University
Student Paper Session
Ajay Bandi ..... Northwest Missouri State University
Scott Sigman ..... Drury University
Student Poster Competition
Joseph Kendall-Morwick ..... Park University
Dayu Wang ..... St. Charles Community College
Student Programming Contest
Charles Riedesel ..... University of Nebraska-Lincoln
Dayu Wang ..... St. Charles Community College

Chris Branton ..... Drury University
Amos Gichamba ..... Southwest Baptist University

Two-Year College Outreach

Rex McKanry ..... St. Charles Community College
Belinda Copus ..... University of Central Missouri

Local Arrangement (Career Fair/Local Vendors)

Carol Browning ..... Drury University
Chris Branton ..... Drury University
Shannon McMurtrey ..... Drury University
Scott Sigman ..... Drury University

Zoom Committee

Brian Hare ..... University of Missouri Kansas City
Scott Sigman ..... Drury University

Regional Board — 2022 CCSC Central Plains Region

Regional Rep & Board Chair

Judy Mullins ..... Retired

Registrar & Membership Chair

Ron McCleary ..... Retired

Current Conference Chair

Scott Sigman ..... Drury University

Next Conference Chair

Mahmoud Yousef ..... University of Central Missouri

Past Conference Chair

Brian Hare ..... University of Missouri Kansas City

Secretary

Diana Linville ..... Northwest Missouri State University

Regional Treasurer

Denise Case ..... Northwest Missouri State University

Regional Editor

Joseph Kendall-Morwick ..... Park University

Webmaster

Michael P. Rogers ..... University of Wisconsin Oshkosh

Reviewers — 2022 CCSC Central Plains Conference

Rad Alrifai ..... Northeastern State University, Tahlequah, OK
Beth Arrowsmith ..... University of Missouri - St. Louis, Saint Peters, MO
Ajay Bandi ..... Northwest Missouri State University, Maryville, MO
Chris Branton ..... Drury University, Springfield, MO
Kevin Brunner ..... Graceland University, Lamoni, IA
John Buerck ..... Saint Louis University, St. Louis, MO
David Bunde ..... Knox College, Galesburg, IL
Chia-Chu Chiang .... University of Arkansas at Little Rock, Little Rock, AR
Chris Cox ..... Westminster College, Fulton, MO
Anurag Dasgupta ..... Valdosta State University, Hahira, GA
George Dimitoglou ..... Hood College, Frederick, MD
Russell Feldhausen ..... Kansas State University, Shawnee, KS
Ernest Ferguson ..... Northwest Missouri State University, Maryville, MO
David Furcy ..... University of Wisconsin - Oshkosh Oshkosh, WI
Suvineetha Herath ..... Carl Sandburg College, Galesburg, IL
James Jones ..... Graceland University, Lamoni, IA
Brian Kokensparger ..... Creighton University, Omaha, NE
Eric Manley ..... Drake University, Des Moines, IA
Chris Mayfield ..... James Madison University, Harrisonburg, VA
Thomas Mertz ..... Kansas State Polytechnic, Salina, KS
Jose Metrolho ..... Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
Muath Obaidat ..... LaGuardia Community College, Long Island City, NY
Kian Pokorny ..... McKendree University, Lebanon, IL
Hassan Pournaghshband ..... Kennesaw State University, Kennesaw, GA
Charles Riedesel ..... University of Nebraska - Lincoln, Beatrice, NE
Jamil Saquer ..... Missouri State University, Springfield, MO
Cecil Schmidt ..... Washburn University, Topeka, KS
William Siever ..... Washington University, St. Louis, MO
Timothy Urness ..... Drake University, Des Moines, IA
Henry Walker ..... Grinnell College (retired), Napa, CA
Maria Weber ..... Saint Louis University, St. Louis, MO
Linda Webster ..... Westminster College, Fulton, MO
Cong-Cong Xing ..... Nicholls State University, Thibodaux, LA
Baoqiang Yan ..... Missouri Western State University, Saint Joseph, MO

A Software Engineering Career from the Perspective of Availability, Reliability and Maintainability*

Friday Opening Keynote

Michael Newton
Software Scientist/Chief Engineer, L3Harris Technologies

Abstract

Many software engineering students graduate from college prepared to develop software, but, unfortunately, they lack many fundamental career skills necessary to function on a software engineering team. Writing code is just one small aspect of a software engineering career. Many foundational work skills and habits are necessary to be a productive software engineer, and these skills and habits can be loosely categorized into one of the following system design attributes of Reliability, Availability, and Maintainability:

- Reliability as it equates to career maturation, teamwork, and work ethic
- Availability as being pro-active with task completion and time management behavior
- Maintainability for keeping communication, software, and career skills relevant

The goal of this presentation is to provide computer science instructors and students real life examples and guidance on these three attributes.

Bio

Michael Newton is a Software Scientist at L3Harris Technologies leading system engineering and software engineering teams in the development of mission system software. He is a graduate of Southwest Baptist University with a B.S. in Computer Science/Mathematics and a M.S in Software Engineering from Texas Christian University. His career includes technically leading geographic disparate, embedded software engineering teams at Motorola Solutions for 28 years. In addition, he taught Computer Science courses from a practitioners perspective at Southwest Baptist University during the 2002-2003 school year.

How the Pandemic Has Change Talent Management*

Banquet Keynote

Mark Garton
Senior Director of IT Governance and Data Strategy,
O'Reilly Auto Parts Stores, Inc

Abstract

Hiring and retaining talent is more challenging than ever before. The Pandemic has changed team dynamics and interactions with team members for many companies. These new realities have forced managers to adjust how they engage with their teams. It has also changed the skills needed to succeed in a complex corporate environment. It takes more than technical skills to be successful. The goal of the presentation is to share strategies used to find, develop, manage and retain talent in today's ever-changing job market.

Bio

Mark Garton is the Senior Director of IT Governance and Data Strategy at O'Reilly Auto Parts Stores, Inc. He leads teams responsible for information security, software quality assurance, and data strategy. With over 24 years of experience as an IT professional, he has worked in information technology from software development to establishing a cyber security program. He is passionate about developing leaders and created a leadership development program for new technology managers. He holds a BA in Mathematics from Drury University and a Master's Degree in Computer Information Systems from Missouri State University.

How To Secure ABET Accreditation for a Cybersecurity Program: A Case Study*

Xiaodong Yue, Belinda Copus, and Hyungbae Park
Department of Computer Science
University of Central Missouri
Warrensburg, MO 64093
{yue, copus, park}@ucmo.edu

Abstract

In response to the shortage of a qualified cybersecurity workforce, many new cybersecurity programs were implemented recently across the country. Since it is a new discipline, curriculum guidelines and ABET accreditation criteria for Cybersecurity program had to be developed[2]. As of October 2021, there were only 13 ABET accredited baccalaureate Cybersecurity programs worldwide[1]. In this paper, a case study is presented based on the lessons and experiences learned from the most recent successful ABET accreditation preparations for our undergraduate Cybersecurity program. Some of the best practices summarized in this paper are proven to be useful and practical, and can be adopted in a similar setting by other institutions especially those with no or little ABET accreditation experience.

1 Introduction

The US Commission on Enhancing National Cybersecurity-a non-partisan commission charged with developing recommendations to ensure the digital economy's growth and security released the "Report on Securing and Growing the Digital Economy" in 2016[6]. The report presents data on the US's and global economy's shortage of cybersecurity professionals and practitioners and recommends efforts be expanded to attract and train more workers.

Colleges and universities across the country are launching initiatives to establish new cybersecurity programs or courses of study within existing computing programs. In Missouri alone, there have been 7 new undergraduate and 6 new graduate Cybersecurity programs started since 2010.

With the creation of new degree programs in Cybersecurity, there is an ever increasing demand for curricular guidance in cybersecurity education and accreditation criteria to facilitate programs to delivery contents not only to meet the workforce needs but also to maintain the quality and rigor. In response to such demand, the Cyber Education Project (CEP) a consortium of institutions with interests in improving cybersecurity education was formed in 2014[4]. Computing Accreditation Commission of (CAC) ABET developed the accreditation criteria for cybersecurity program based on the final report from CEP. Further refinement of Cybersecurity curriculum was introduced in 2017[5]. At the time of this writing, Engineering Accreditation Commission (EAC) of ABET also has developed accreditation criteria for cybersecurity engineering programs. This paper will focus on the CAC accreditation criteria under which our program is currently accredited.

The rest of the paper is organized as follows. Section II provides background information for our work. Detailed information concerning how to meet eight ABET accreditation criterions are described in Section III. Conclusions are presented in Section IV.

2 Background

The University of Central Missouri (UCM) is a public comprehensive regional university with enrollment of over 10,000 students. The Cybersecurity program is housed in the School of Computer Science and Mathematics. Besides the ABET accredited Cybersecurity program, the school also offers an ABET accredited Computer Science program and an ABET accredited baccalaureate degree program in Software Engineering. Master's degree programs are also offered in Computer Science, as well as in Cybersecurity and Information Assurance.

As of Fall 2021, there are 21 full time computer science faculty and two full time staff in the School of Computer Science and Mathematics. Out of the 21 full time faculty, 13 are tenured or tenure track faculty, the rest are non-tenure track faculty. There are 160 students enrolled in the undergraduate Cybersecurity in Fall 2021. The ABET accreditation timeline for the UCM Cybersecurity program is listed below.

- Program started in Fall 2014
- Request for Evaluation (RFE) sent to ABET in January 2017
- First graduate from the program in May 2017

- Self-study report submitted to ABET in June 2017
- CAC site visit in October 2017
- CAC draft statement received in January 2018
- Official accreditation decision received in August 2018
- CAC re-accreditation site visit in October 2019 (to synchronize accreditation visits with other CAC accredited programs on campus)
- Official re-accreditation decision received in August 2020

It is also worth mentioning that our cybersecurity program self-study report was selected by ABET as an example of well-written and well-organized reports to be displayed at the 2018 ABET Symposium.

3 Meeting the Criteria

Any program accredited by CAC of ABET need to meet the 8 general criteria and the program criteria if applicable[3]. Next, we will go through each of the criteria one by one to provide a synopsis on our understanding of the key issues to be addressed and areas needing special attention.

3.1 Criterion 1: Students

This criterion focuses on how students are to be evaluated, monitored, and advised by the program. The program must define and enforce policies on student admission, transfer credits and graduation requirements.

The first key issue in this criterion is student advising. Our university adopts a dual advising model. Each student has a designated academic advisor and a faculty advisory. Students will voluntarily receive advising from the College Advising Center from an academic advisor. Since the criterion mandates that "students must be advised regarding curriculum and career matters", our program fulfills this requirement by requiring each cybersecurity major to meet their faculty advisor at least once per semester usually before the registration starts for next semester. A school advising hold is placed on the student account one month prior to the opening of course registration and is removed after the student meets with their advisor. School advising begins during the first semester of attendance for students in the Cybersecurity program. The advising focuses on courses in the area of Cybersecurity/Computer Science, Mathematics, and Networking. During this advising period, in addition to course planning questions, students are able to request general advice regarding internships and future job opportunities in their chosen area.

The second key issue in Criterion 1 is how to evaluate and award transfer credits. The program must define and enforce policies for awarding appropriate academic credit for courses taken at other institutions. At UCM, upon transfer,

a student's transcript(s) of prior coursework is evaluated. An official transfer table is referenced to determine if a student's completed course will transfer. Occasionally, a student will have taken a course at their previous institution that is not listed in the transfer table. In Cybersecurity, courses that do not appear in the transfer table, and are requested for review by the student, are evaluated by a member of the Cybersecurity/Computer Science faculty as directed by the School Chair. This faculty member will have expertise in the respective technical area and will examine the syllabus and grade received for the course. A recommendation is made to the chair to accept or deny the course credit. Transfer courses are accepted if they substantially meet the requirements of the corresponding UCM Cybersecurity/Computer Science course. A similar policy is also in place to evaluate and award transfer credits from a foreign institution.

The third key issue in this criterion is that the program must define and use a policy to document any exceptions made to a student's program of study and graduation requirements. It is highly recommended that exceptions are approved only in special circumstances. Frequent exceptions without proper documentation could be a major concern if the program evaluators identify issues during the transcript analysis. Finally, the program must make sure that the degree name displayed on the official transcript matches the degree name seeking ABET accreditation. This is especially important for programs which only seek accreditation for one of its options.

3.2 Criterion 2: Program Educational Objectives

This criterion requires a program to have a documented, systematically utilized, and effective process for the periodic review of program educational objectives (PEOs).

The first key issue in Criterion 2 is that it requires that the program's constituents be involved in the review and revision of the PEOs. Each program has the freedom to determine who the constituents are. Our program's constituents are employers, current program students, program alumni and program faculty, which also form the membership of the program's Advisory Board.

The second key issue in this criterion is that a program must have a process to periodically and systematically review PEOs. Although ABET does not specify how frequent a program's PEOs need to be reviewed, it is a good practice to review them on a regular basis to ensure they remain consistent with the institutional mission and the program's constituents' needs. Our program's process is that faculty review the PEOs annually the semester before the advisory board meeting. Any proposed changes to PEOs are voted by all the faculty. The advisory board meets a semester later to review and discuss

the proposed revisions to PEOs and present any recommendations that they might have.

Third, a program must ensure its PEOs are consistent with the mission of the institution. We'd suggest using a table to map each of the program PEOs to the corresponding mission statement. In addition, PEOs are broad statements that describe what graduates are expected to attain within a few years of graduation and are distinguishable from Student Outcomes. PEOs which could be interpreted as student attainment by graduation should be avoided. PEOs that do not meet this requirement will be considered a Criterion 2 shortcoming.

Finally, if the cybersecurity program is housed within a department containing multiple programs, engaging constituencies at a department level or using department-level processes can result in overlooking the needs of the cybersecurity program, especially if that department is dominated by another program or has programs accredited by another commission. For our case, our cybersecurity program has a completely different constituent and process from computer science and software engineering programs. In addition, the PEOs must be published in a manner that makes them available to the program's constituencies. Our cybersecurity program's PEOs are published in the university catalog and publicly accessible program website.

3.3 Criterion 3: Student Outcomes

The most recent ABET accreditation criteria for Cybersecurity has 6 required student outcomes. A program may define additional outcomes. Since the ABET student outcomes are fairly comprehensive, the faculty decided that no additional outcome was needed for our program. Since all the student outcomes are required, ABET no longer requires periodic review of student outcomes. Finally, it is worth mentioning that the number of student outcomes correlates the amount of assessment efforts in the next criterion. A program is highly recommended to use its best judgement on adding additional outcomes. Similar to PEOs, student outcomes must be published on university catalog and/or program website.

3.4 Criterion 4: Continuous Improvement

This criterion, in our opinion, is the center piece for the whole ABET accreditation process. It focuses on the processes for assessing and evaluating attainment of student outcomes as well as the continuous improvement of the program.

Our assessment process starts with the mapping of courses in the program to the 6 student outcomes. Faculty then determine which courses are most

appropriate to assess each outcome. In order to provide a reliable evaluation on the extent to which the student outcomes are being attained, our program assesses each outcome in at least two courses preferably at the upper level. Each of the student outcomes have been defined by a few (typically 2-4) high-level performance indicators so that they can be integrated into the curriculum and measured in a consistent and reliable manner. The performance indicators allow instructor discretion in selecting particular assignments, test questions, and other metrics to quantify assessment of the outcome. In addition, rubrics have been created for each performance indicator to measure the attainment level of an outcome. Student artifacts (student work) are given a ranking of "does not meet the expectation", "meets the expectation" or "exceeds the expectation" according to the rubrics. The rankings of the work produced by the students are then totaled to determine what level of attainment is being made toward a student outcome. A goal of 75% or higher was established by the faculty to determine if attainment of a student outcome is being met at the course level. The goal attainment metric is calculated by summing the percentages in the "meets" and "exceeds" categories. Attainment of an outcome can be further analyzed by reviewing all of the performance indicators for an outcome across courses in which the outcome is assessed and determining the level of attainment for the outcome. Indirect assessment instruments such as senior exit surveys are also used by the program to assessment the attainment of student outcomes.

The continuous improvement process for the cybersecurity program involves assessing the degree of attainment of the student outcomes; evaluating the assessment results; identifying improvement needs and opportunities; and implementing the indicated program improvements. It is highly recommended that a mechanism to keep track of all issues identified in the continuous improvement process to be utilized ensure that they are ultimately resolved. In our program a master spreadsheet is used to track open issues.

It is worth noting that when collecting data from a course for assessing a student outcome it is expected that the data will be separated by program, as our focus is about the performance of students enrolled in the cybersecurity program. Combining data from students in different majors would mask potential issues related specifically to the Cybersecurity program.

The "Definitions" section of the Criteria makes a distinction between "assessment" and "evaluation." These are separate processes. The Criteria calls for using "appropriate" processes, not the "most efficient," nor the "most effective," for assessing and evaluating the extent to which the student outcomes are being attained. The test here should be whether the process (1) is sufficient to allow the program to make informed judgments, and (2) is sustainable by, and appropriate to the needs of, the program.

3.5 Criterion 5: Curriculum (including program criteria for cybersecurity)

This criterion, along with the cybersecurity program criteria, details the CAC cybersecurity curriculum requirements. ABET cybersecurity curriculum requires at least 45 semester credit hours of computing and cybersecurity course work and at least 6 semester credit hours of mathematics that must include discrete mathematics and statistics. Our program offers two concentration areas which both meet all the specific curriculum requirements. Particularly, 42 credits in cybersecurity course work and 12-18 credits in computing. It is worth noting that, unlike computer science, the cybersecurity curriculum criteria do not specify that the mathematics component must have mathematical rigor at least equivalent to introductory calculus.

It is worth noting that this criterion has an explicit statement that curriculum requirements do not prescribe specific courses. Programs need to show where the coverage is, but this portion of the criterion can be satisfied by indicating in what course(s) each topic is covered. Table 1 shows the required courses from the UCM's cybersecurity program which provide coverage on various required cybersecurity topics.

Table 1: UCM's Required Cybersecurity Courses

Course	**Title**	**Credits**
CS 1030	Intro. to Computer Programming	3
CYBR 1500	Command Line Environments	3
CYBR 1800	Intro. to Cybersecurity	3
CYBR 2500	Computer Systems Administration	3
CYBR 3130	Secure Programming	3
CYBR 3300	Intro. to Cryptography	3
CYBR 3510	Systems Security	3
CYBR 3520	Intro. to Cyber-Physical Systems Security	3
CYBR 3820	Usable Privacy and Security	3
CYBR 3830	Economics of Cybersecurity	3
CS 3840	Computer Networking	3
CYBR 4820	Intro. to Information Assurance	3
CYBR 4840	Ethical Hacking	3
CYBR 4850	Computer and Network Forensics	3
Additional Electives in Computing Course		10-24
Total		52-66

3.6 Criterion 6: Faculty

One of the main focuses on this criterion is faculty competency. Our program strives to provide plenty of professional development opportunities to the faculty. Funds are allocated to support faculty travel for conferences/workshops, etc. Furthermore, all UCM faculty who are primarily responsible for cybersecurity instruction have at least one industrial cybersecurity certificate such as GPEN, GCFE, GNFA, etc. Faculty are also encouraged to participate in training through UCM Center of Teaching and Learning, webinars and/or other free professional development opportunities.

A program must have sufficient number of faculty to provide instruction, student advising and program management. It is highly recommended that at least two faculty members are available to teach each course in the entire curriculum and faculty are given sufficient training to enable effective and efficient student advising.

There are no specific guidelines for faculty teaching loads in cybersecurity and many factors influence what is a reasonable teaching/research load for faculty. Since there are requirements in the Faculty Criteria that the faculty must be engaged in professional development, student advising and other standard faculty activities, the general rule is that teaching more than 12 credit hours per semester sometimes have issues appropriately engaging in all of those required faculty activities. At UCM, all tenured/tenure track faculty have a 9 credit hours teaching load per semester while non-tenure track faculty's load is 12 credit hours per semester.

3.7 Criterion 7: Facilities

This criterion is mostly resources related. With the budget cut in public higher education recently, it is a challenge for many programs to systematically maintain and upgrade labs, classrooms, and other critical instructional resources. At UCM, this problem is diminished by including a student tech fee and through revenue sharing generated through course offerings at extended campuses. Faculty are also encouraged to use open source software for instruction as long as it does not compromise the quality of the knowledge content. In addition, the School has worked closely with the University Foundation to seek private donations. A major gift was received several years ago to renovate our student study area and a computer lab. Nevertheless, a program still needs to have other innovative approaches and/or work closely with the administration to secure funds to maintain and upgrade its facilities and resources.

3.8 Criterion 8: Institutional Support

This criterion emphasizes on adequate institutional support and leadership to ensure the quality and continuity of the program. We are very fortunate to have an immensely supportive administration regarding our endeavors for ABET accreditation. It is infeasible for a program to secure and/or maintain ABET accreditation without support from the administration. Finally, it is worth noting that a common misunderstanding is that a program must have its own dedicated IT tech staff. In fact, it is unnecessary as long as adequate IT support is provided at the college and/or university level.

It is also worth noting that a program should carefully review the extent to which faculty needs are met by the institution's computer procurement policy. For example, a centralized, "one-size-fits-all" approach or a "one computer per faculty member" limit might not provide sufficient computational resources for one's continued professional development.

4 Conclusions

In summary, our ABET accreditation efforts have been very rewarding. It has not only added the value to the degrees our students receive but also aids in student recruitment. We hope that other cybersecurity programs considering ABET accreditation will benefit from the experiences discussed in this paper.

References

[1] ABET accredited programs inventory. `https://amspub.abet.org/aps/category-search`. Accessed October 2021.

[2] ABET approves accreditation criteria for undergraduate cybersecurity programs, 2018. `https://www.abet.org/abet-approves-accreditation-criteria-for-undergraduate-cybersecurity-programs/`.

[3] Criteria for accrediting computing programs, 2021. `https://www.abet.org/wp-content/uploads/2021/01/C001-21-22-CAC-Criteria.pdf`.

[4] Final Report of the Cyber Education Project (CEP) Accreditation Committee. Cyber education project. pages 1–16, 2016.

[5] Joint Task Force on Cybersecurity Education. Cybersecurity curricula 2017: Curriculum guidelines for post-secondary degree programs in cybersecurity. New York, NY, USA, 2018. ACM.

[6] The US Commission on Enhancing National Cybersecurity. Report on securing and growing the digital economy. 2016.

A Literature Review on User Acceptance of AI-Enabled Application*

Gary Yu Zhao[1] *and Cindy Zhiling Tu*[2]
[1]*College of Business & Information Systems*
Dakota State University
Madison, SD 57042
Gary.Zhao@trojans.dsu.edu
[2]*School of Computer Science and Information Systems*
Northwest Missouri State University
Maryville, MO 64468
cindytu@nwmissouri.edu

Abstract

This study focuses on individuals' acceptance of AI-enabled applications. Through a thorough literature review, this study proposes a theoretical model to identify factors affecting individual users' acceptance of AI applications

1 Introduction

Artificial intelligence (AI) is commonly defined as the information technology (IT) capabilities that can perform tasks that possibly require intelligence [21]. Prevailing AI technologies, including machine learning, speech recognition, facial recognition, robots, self-drive vehicles, natural language processing (NLP), and virtual agents, are being deployed to a huge variety of existing systems and new applications. In addition, the increasing availability of big data, growing computing power, and advanced machine learning (ML) algorithms have led to an astounding development of AI-enabled applications (AIapps). As a result, people's lives in different areas, including their homes, education, healthcare,

employment, entertainment, safety, and transportation, are re-shaped dramatically by AI applications [2].

When AIapps are combined with personal devices such as smartphones, tablets, smartwatches, and the IoTs, people's daily lives have been transformed even without realizing it. Today, AIapp via emotion-sensing facial recognition can detect if a person is upset, sad, annoyed, or happy, and it is used for improving customer satisfaction [12]. AIapps with voice queries and NLP like Amzaon Alexa (on Amazon Echo), Siri (iPhone, iPad, iOS laptop), Google Assistant (Google phone, Google Home, Hyundai car), Cortana (Microsoft phone, Windows platform) can help people make calls, send messages, answer questions, provide recommendations, set the alarm, make a to-do list, play music and provide real-time information on weather, traffic, news, sports and more. AIapps with NLP like Replika (smartphone app) and Wysa (a chatbot) can talk with people like a friend. AIapps with image recognition like FaceApp (a smartphone/tablet app) can help people convert an actual picture into an amazing one. ELSA Speak with NLP can help people learn to speak English. Socratic with image recognition can assist students with their homework just by submitting a picture of the tasks. In addition, many AIapps are running at the backend, which the users do not perceive. Netflix movie recommendation system is a typical example of this kind of AIapp. Many AIapps that were thought impossible before are now becoming true.

The market of AIapps is growing fast. Nonetheless, even the AIapps are conveniently accessed from personal devices and are free to use, and most people do not use them regularly. For example, a recent survey showed that 98% of iPhone users had used Siri in the past. However, only 30% used it regularly, with 70% rarely or only occasionally using it [5]. How attractive is AI to individual users? Why do people tend to or refuse to use AI-enabled applications? These are interesting research topics. However, AI researches have been conducted on the technology, advantages, side effects, limitations, and its forthcoming impact on society. Very little research has focused on individuals' acceptance of AI technology. Due to its unique characteristics, AI has introduced new features to the traditional technology acceptance model. Therefore, it is valuable to understand individual users' adoption behavior in the new AI context. Through a thorough literature review, this study proposes a theoretical model to identify factors affecting individual users' acceptance of AIapps, which has seldom been empirically studied in the literature.

2 Literature Review

To locate relevant literature, we conducted keyword and abstract searches from four prominent online databases EBSCOhost, ProQuest (ABI/INFORM), Sci-

enceDirect (Elsevier), and ACM Digital Library. We created three groups of keywords: #1, ("AI-enabled application" OR "AI-powered application" OR "AI-based application" OR "AI application"); #2, ("adoption" OR "acceptance"); #3, ("consumer" OR "individual users" OR "personal devices" OR "mobile devices"). For each database, we conducted three rounds of searches, i.e., first round (using only #1 keywords), second round (using #1 and #2 keywords), and third round (using #1 and #2 and #3 keywords). The filters for three rounds were the same, "year, 2017-2022", "research articles", "subject area, Computer Science, Business, Management and Accounting, Social Science". In total, 242 articles were extracted from the online database. Then, we manually scanned abstracts and filtered out those less relevant articles focusing on AI application development, AI algorithm, gov/industry usage, patent-related, enterprise usage, user usability related, etc. Finally, twenty-one peer-reviewed academic articles were selected for the literature review.

2.1 Common features of AI-enabled applications

AIapps have been deployed in most aspects of people's daily lives. So what are the common features of AIapps, and how do they attract people to use them? Five general attributes are brought out from the collected academic articles.

First, machine learning capability. Scholars acknowledge that machine learning (ML) ability is one of the most outstanding features of AIapps [1, 3, 9, 14, 15]. AIapps must have the ability to continuously learn through data and experience to adapt to their environment [2]. Ruiz-Real et al. (2021) argue that AI-enabled systems with ML ability have a common application in the big data analyzing field, such as a complicated recommender system based on an enormous volume of inputs [20].

Second, human-like interactivity. AIapps must have the ability to interact with people in a natural way [1]. Recent developed Natural language processing and understanding (NLP/NLU) has already been deployed to a vast majority of daily applications such as service chatbots, Siri, Google Assistant, Amazon Alexa, etc. These AIapps can interact with the user as a human [9, 10, 15]. A human-like chatbot with the anthropomorphic quality should be able to respond to the user based on the keywords, determine what type of problem is faced by the customer, understand the user's attitude and emotion, predict the feedback of the user, and try to pacify a frustrated user [3, 4, 7, 16, 22]. Also, a human-like voice AI assistant can be perceived as a friend. This relationship between the user and an AI assistant brings a sense of social presence to user's mind, following building a rapport with the AI agent [4, 16].

Third, knowledge representation and reasoning. Reasoning is always associated with human intelligence. Previous efforts in AI were focused on creating an application that could reason by itself, making conclusions from some

premises [15]. Many AIapps such as digital assistants or chatbots are a kind of knowledge-based application, and they can search, extract, analyze, and represent the knowledge [9, 10, 20]. An AIapp must be able to absorb, store, transform, process the data from both new and existing sources and represent that into the system using effective models and schemas [3, 17]. Moreover, an AIapp should have abilities to draw inferences from provided data [1, 4].

Forth, Intelligent algorithm. AIapps are based on rules and algorithms that can be applied to variant areas and outperform human-level intelligence [9]. An AIapp must have the capability of computation and pattern recognition using provided data [15]. Canhoto & Clear (2020) state that one of the AI characteristics is its cognitive capacity that use ML algorithm to detect patterns in the input data, learn from mistakes, and self-correct [3]. An AIapp must be associated with human-level intelligence such as statistical analysis, ML, classification, optimization, ranking/sorting, generating hypotheses based on confidence interval [14].

Fifth, Autonomy. Contemporary forms of AIapps keep increasing their ability to act independently without human intervention [2, 15]. AIapps eliminate the human emotional component and the flexibility of thought and actions not following the strict rules. This autonomy attribute of AIapps allows them to process difficult problems beating humans by miles [20]. Many AIapps (Ad pusher scripts) run unnoticed by the end-users to help them improve their performance even change their lives [3].

These five characteristics are common features of AIapps. However, many scholars claim that some other unique or more advanced AIapps or platforms could have more features such as big data processing ability, pattern recognition, relationship perceiving, etc. Furthermore, all these natural features of AIapps bring practical functions and valuable benefits and attract people to use them.

2.2 Technology Acceptance Model

The Technology Acceptance Model (TAM) is proposed by Davis (1989) based on the Rational Action Theory (TRA). TAM has been used to explain the users' adoption of new technology or services in many fields, especially in the information system discipline. Davis (1989) provides a theoretical framework for researchers to understand what external factors affect users' intention to accept the new technology. TAM originally summarizes the variant factors into two: perceived usefulness (PU) and perceived ease of use (PEU) [6]. Although it continues to be extended to upgraded models with later research such as TAM 2, TAM3, UTAUT, etc., TAM is more popular than the others and focuses on the major determinants of user acceptance. In addition, Davis (1989) possesses consistent measurement tools and explains the significant variance in adoption

intentions [6]. Moreover, TAM has been used by many scholars and it provides a big volume of questions for each factor, increasing reliability to the relevance of the measurement. In this research, TAM is used as a core model to understand the users' adoption of AIapps. Though TAM is an efficient tool to explain the acceptance of new technology, the extended variables related to AIapps must be considered to understand the acceptance distinctly [13].

2.2.1 Perceived usefulness (PU)

Davis (1989) defined perceived usefulness as "the degree to which a person believes that using a particular system would enhance their job performance" [6]. PU affects users' acceptance positively. Machine learning capability, knowledge reasoning ability, and intelligent algorithms enable AIapps to meet people's expectations and satisfy people's demands. In other words, AIapps are useful and bring benefits to humans.

Nearly all scholars acknowledge that AIapps bring enhancement in people's lives profoundly and increase the satisfaction and efficiency or fun in a variety of people's social activities [1, 2, 8, 9, 10, 15, 17, 20]. For example, a movie recommender based on machine learning technology can provide users with new movies by learning users' preferences and other people's reviews, which makes people feel satisfied with saving time of choosing movies from thousands of items in the pool. If people believe they can gain from AIapps, they are more likely to use them. These gains include improving the users' social interaction, personal identity, the extension of conformity, saving time, improving life efficiency [4, 24]. Also, McLean et al. (2020) demonstrate that Alexa can help people complete tasks more efficiently and quicker than make people's lives easier, more convenient, providing social presence and enjoyment [16].

Providing high-quality information and services based on an intelligent algorithm and machine learning is important perceived usefulness of AIapps for users [7, 11, 13]. Knowledge reasoning ability enables AIapps to provide people with causation analysis, predictions, and summary information, outperforming people's information processing capabilities [9]. Also, AIapps can handle big data to provide users with personalized and customized information [9].

2.2.2 Perceived ease of use (PEU)

Davis (1989) defined perceived ease of use as "the degree to which a person believes that using a particular system would be free from effort" [6]. PEU affects users' acceptance positively.

AIapps change traditional human-machine interaction with no need for physical input, i.e., hands-free instead of typing, clicking, and navigation interface [16]. Furthermore, the human-like interaction makes people believe that

AIapps is easy to learn and easy to use. People tell a voice assistant like Siri, Google Assistant, and Alexa what they want to do and use natural language to communicate with a text-based service chatbot. These usages do not need people's extra efforts to learn how to use AIapps [7, 11, 18]. In addition, people prefer using the technology if the devices or systems already exist, e.g., an AI-enabled shopping chatbot has been added to Facebook, WhatsApp, or WeChat, quickly start to use without hassle [13].

The characteristic of autonomy enables AIapps to serve people independently even without people's perceptions, which reduces the effort of use to a huge extension. When AIapps are easier to use, people will have more self-efficacy, self-control, and the positive intention to accept them [24].

2.2.3 Extended variables

Perceived trust risks. People always view AIapps as black-boxes and hardly trust them [9]. There are two reasons mainly. First, it is hard to understand the technology of machine learning, and how it works in AIapps. Second, AIapps have the autonomy that they can run by themselves or even are invisible for most people. Trust issues include users' concerns over algorithmic non-transparency, online vulnerabilities, immature technology, bias and uniqueness neglect, social classification, delegation, the privacy of their interactivities, and the potential for private information to be uncovered to third parties [9, 10, 16, 17, 18, 19]. About 80% of researchers claim that the number one perceived issue is trust risk that plays a negative role in the adoption of AIapps. Further, Causable explainable AIapps can help users understand how AI algorithms work and increase users' trust, which leads to more usage [23].

Perceived enjoyment or hedonic value. AIapps with high intelligence can help people complete complicated tasks by simplifying, optimizing, and customizing the processes, which makes people relax, feel happy and avoid frustration [8]. Users intend to use an AIapp if they get enjoyment or hedonic value from it [8, 14, 16, 18]. According to Choi & Drumwright, Fernandes & Oliveira, and Kasilingam, perceived enjoyment is one of the most critical factors to affect the acceptance of AI-enabled voice assistants and chatbots [4, 7, 13].

Subjective Social norms. TAM 2 includes social norms as an important construct that determine the adoption of new technology. People always feel social pressure from their friends, parents, spouses, classmates, etc., when they decide to use an AIapp or not. Stronger positive social norms increase people's intention of using a new AIapp [7, 24].

Perceived behavioral control or self-efficacy. People usually intend to adopt a new AIapp if they feel comfortable controlling the required resources such as time, money, personal capabilities [24]. The users' self-efficacy is the perceived ability to control the environment to achieve a particular goal, and it is an

important factor that positively affects the adoption of AIapps [17].

3 Theoretical Development

We developed our research framework by integrating the main concepts reflected in the literature (see Figure 1).

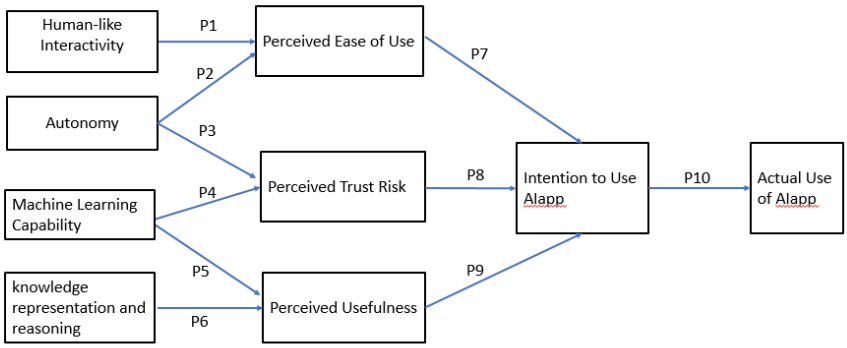


Figure 1: Proposed Research Model

First, we extend the TAM model by adding a new construct that is perceived trust risk. Three independent variables affect the intention to use AIapp, and the intention leads to the actual use of AIapp. Then, we examine and map the relationships between four unique features of AIapp and perceived usefulness, perceived ease of use, and perceived trust risk. As most of the researchers agreed, the machine learning capability is the most important characteristic of AIapp, and it allows AIapp to provide useful services by continuously learning new knowledge and adapting itself to the new environment [2]. However, it is the learning skill of a computer that brings people beyond understanding that leads to the trust problem [9]. The ability of knowledge representing and reasoning allows AIapp to provide users high-quality services such as data summary, analysis, and prediction, which brings high-level effectiveness and efficiency to people's tasks [7, 11, 13]. AIapp with human-like interactivity allows users to communicate with the machine using natural languages in voice or typing as well as body gestures, which reduces the effort of learning and using the application. Current AIapps run without human intervention or even without people's perceptions [2, 15]. This autonomy feature allows users to use the application easily but simultaneously leads to the trust issue. Therefore, based on the relationships between unique features of AIapp and three

constructs of the TAM model, we propose that:

P1: The feature of human-like interactivity positively influences users' perceived ease of use.

P2: The feature of autonomy positively influences users' perceived ease of use.

P3: The feature of autonomy positively influences users' perceived trust risk.

P4: The feature of machine learning capability positively influences users' perceived usefulness.

P5: The feature of machine learning capability negatively influences users' perceived trust risk.

P6: The feature of knowledge representing and reasoning positively influences users' perceived usefulness.

P7: Perceived ease of use positively influences user's intention to use AIapps.

P8: Perceived usefulness positively influences user's intention to use AIapps.

P9: Perceived trust risk negatively influences user's intention to use AIapps.

P10: User's intention to use AIapps positively influences user's actual use.

User demographic characteristics such as age, gender, location, and education may be used as control variables when testing the research model.

4 Discussion and Conclusion

This study focuses on what factors determined by the common features affect individual users' intention to use AIapps. First, limited literature summarized the common features of AIapps, which are the root causes of the factors affecting the adoption of AIapps. We summarized five characteristics of AIapp, i.e., machine learning capability, human-like interactivity, knowledge representing and reasoning, intelligent algorithms, and autonomy. A map between the factors and the features helps better understand how attractive the AIapps are and determine new factors in the later study. It brings implications both on theoretical and practical aspects to tease out relationships between features of AIapps and the factors that affect the adoption of AIapps.

Second, TAM is the core model for studies of the acceptance of AIapps. Based on TAM, we categorize the factors discussed in previous literature into three: the factor of perceived usefulness, the factor of perceived ease of use, and the others. Other than these two original TAM factors, perceived trust risk is the most proposed factor negatively affecting acceptance. We integrated these factors into our proposed research model.

Third, only a few researchers demonstrate the relationships between natural features of AIapps and the factors affecting the acceptance. Some of them argue that machine learning ability is the principal characteristic to determine usefulness. Some others argue that human-like activity is the first attribute to affect perceived ease of use. It would be interesting to explore more new features associated with the rapid emergence of AI technology and how these features affect the factors or generate new factors. Future researches can fill this gap by providing a systematic and completed study on these relationships.

This study is expected to contribute to both academics and practice. Theoretically, this study proposes a model to identify factors affecting individual users' acceptance behavior of AI-enabled applications, which so far has seldom been empirically studied in the literature. This model also enriches general TAM by investigating how unique AI features may affect and mediate users' acceptance intention and behavior. Practically, the results of this research will help AI-enabled application developers or vendors better understand individual users' behavior regarding the use of their applications.

References

[1] Steven Alter. Understanding artificial intelligence in the context of usage: Contributions and smartness of algorithmic capabilities in work systems. *International Journal of Information Management*, page 102392, 2021.

[2] Nicholas Berente, Bin Gu, Jan Recker, and Radhika Santhanam. Managing artificial intelligence. *MIS Quarterly*, 45(3):1433–1450, 2021.

[3] Ana Isabel Canhoto and Fintan Clear. Artificial intelligence and machine learning as business tools: A framework for diagnosing value destruction potential. *Business Horizons*, 63(2):183–193, 2020.

[4] Tae Rang Choi and Minette E. Drumwright. "OK, Google, why do I use you?" motivations, post-consumption evaluations, and perceptions of voice AI assistants. *Telematics and Informatics*, 62:101628, 2021.

[5] Benjamin R. Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. "What can I help you with?": Infrequent users' experiences of intelligent personal assistants. MobileHCI '17, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989.

[7] Teresa Fernandes and Elisabete Oliveira. Understanding consumers' acceptance of automated technologies in service encounters: Drivers of digital voice assistants adoption. *Journal of Business Research*, 122:180–191, 2021.

[8] Björn Frank, Boris Herbas-Torrico, and Shane J. Schvaneveldt. The AI-extended consumer: Technology, consumer, country differences in the formation of demand for AI-empowered consumer products. *Technological Forecasting and Social Change*, 172:121018, 2021.

[9] Dhruv Grewal, Abhijit Guha, Cinthia B. Satornino, and Elisa B. Schweiger. Artificial intelligence: The light and the darkness. *Journal of Business Research*, 136:229–236, 2021.

[10] Lukas Grundner and Barbara Neuhofer. The bright and dark sides of artificial intelligence: A futures perspective on tourist destination experiences. *Journal of Destination Marketing & Management*, 19:100511, 2021.

[11] Dogan Gursoy, Oscar Hengxuan Chi, Lu Lu, and Robin Nunkoo. Consumers acceptance of artificially intelligent (AI) device use in service delivery. *International Journal of Information Management*, 49:157–169, 2019.

[12] Michael Haenlein and Andreas Kaplan. Artificial intelligence and robotics: Shaking up the business world and society at large. *Journal of Business Research*, 124:405–407, 2021.

[13] Dharun Lingam Kasilingam. Understanding the attitude and intention to use smartphone chatbots for shopping. *Technology in Society*, 62:101280, 2020.

[14] Amit Kumar Kushwaha, Prashant Kumar, and Arpan Kumar Kar. What impacts customer experience for B2B enterprises on using AI-enabled chatbots? insights from big data analytics. *Industrial Marketing Management*, 98:207–221, 2021.

[15] Fernando Martínez-Plumed, Emilia Gómez, and José Hernández-Orallo. Futures of artificial intelligence through technology readiness levels. *Telematics and Informatics*, 58:101525, 2021.

[16] Graeme McLean, Kofi Osei-Frimpong, and Jennifer Barhorst. Alexa, do voice assistants influence consumer brand engagement? – examining the role of AI powered voice assistants in influencing consumer brand engagement. *Journal of Business Research*, 124:312–328, 2021.

[17] Stefano Puntoni, Rebecca Walker Reczek, Markus Giesler, and Simona Botti. Consumers and artificial intelligence: An experiential perspective. *Journal of Marketing*, 85(1):131–151, 2021.

[18] Alexandra Rese, Lena Ganster, and Daniel Baier. Chatbots in retailers' customer communication: How to measure their acceptance? *Journal of Retailing and Consumer Services*, 56:102176, 2020.

[19] Rowena Rodrigues. Legal and human rights issues of AI: Gaps, challenges and vulnerabilities. *Journal of Responsible Technology*, 4:100005, 2020.

[20] José Luis Ruiz-Real, Juan Uribe-Toril, José Antonio Torres, and Jaime De Pablo. Artificial intelligence in business and economics research: Trends and future. *Journal of Business Economics and Management*, 22(1):98–117, 2021.

[21] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, third edition, 2010.

[22] Ben Sheehan, Hyun Seung Jin, and Udo Gottlieb. Customer service chatbots: Anthropomorphism and adoption. *Journal of Business Research*, 115:14–24, 2020.

[23] Donghee Shin. The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI. *International Journal of Human-Computer Studies*, 146:102551, 2021.

[24] Shu-Mei Wang, Yu-Kai Huang, and Chi-Cheng Wang. A model of consumer perception and behavioral intention for AI service. In *Proceedings of the 2020 2nd International Conference on Management Science and Industrial Engineering*, MSIE 2020, page 196–201, New York, NY, USA, 2020. Association for Computing Machinery.

A Functional Programming Course in Remote Learning Model: An Experience Report*

Fahmida Hamid
Computer Science
New College of Florida
Sarasota, FL 34243
fhamid@ncf.edu

Abstract

This experience report shares the teaching methodology of a Functional Programming course (in Haskell) offered amid the Covid-19 pandemic. In addition to highlighting problem-solving and programming from a functional perspective, the course provides a solid example of modern pedagogical elements, including team-based programming, computational thinking, presentation skills, participation and discussion skills, and a sense of accountability. This report presents the methodologies applied to achieve the objectives in a liberal arts setup and discusses the learned lessons that may help an early-career faculty model an intro-programming course.

1 Introduction

Computer Science students should gain experience with multiple programming languages, tools, paradigms, and technologies and study the fundamental underlying principles throughout their education to develop the adaptation skills on new languages and technologies [3, p. 25]. In addition, modern pedagogy

highlights student engagement in order to promote student learning and to help outreach to diverse populations. This experience report describes an introductory course in functional problem solving that integrates such pedagogical elements as team-based programming, computational thinking, presentation skills, participation and discussion skills, and a sense of accountability.

Motivated by the multiline programming paradigm, our institution follows the imperative-first, object-oriented later policy. CS majors at our institution need to take an elective course in Programming Languages (such as Functional Programming (FP)). The FP course offered here expects a mixed bag of students: a group with no prior programming experiences and another group with modest to an exhaustive level of programming experiences (C, Python, Java, Web-programming). It is challenging yet enjoyable to create an evenly playable ground for everyone in such a setup. In the Spring of 2021, we offered the FP course for our students in a remote model. The course objectives are to teach the central ideas of functional programming: abstraction, recursion, higher-order functions, lazy evaluation, program specification and verification, immutable data types, type inference, currying, and programming with combinators. The class practiced active learning [2] strategy: short lecture sessions, intermittent breaks with brain-storming questions, discussions, and dedicated lab hours. The teaching methodology is similar to the Human-Centered Learning (HCL) [6] model. While an HCL model uses section leaders for teaching small interactive learning sessions, this course had the instructor playing multiple roles: instructor while delivering lectures, section leader while working with small groups in labs (virtually), a critique while engaging them in discussions.

The upcoming sections present the overall experiences of handling the related issues in the following order: language and topic selection, mechanics and teaching challenges, students' observations, and a summary of self-reflections.

2 Course Design

A large body of research on teaching introductory programming courses agrees that the primary focus should be on elementary programming concepts, emphasizing developing problem-solving skills [5]. Keeping this in mind, we divided the entire semester into three segments, each focusing on one theme: building basics, introducing advanced topics, and exploring algorithms (figure 1).

Considering Haskell's wide acceptance and applicability, we chose Haskell as the official programming language. However, any other functional programming language such as Scheme or Scala would work. Next, we picked the topics that seem the most appropriate for a novice. One of the most valuable resources for students of the course was the "Learn You a Haskell for Great Good!: A Beginner's Guide" [4] book.

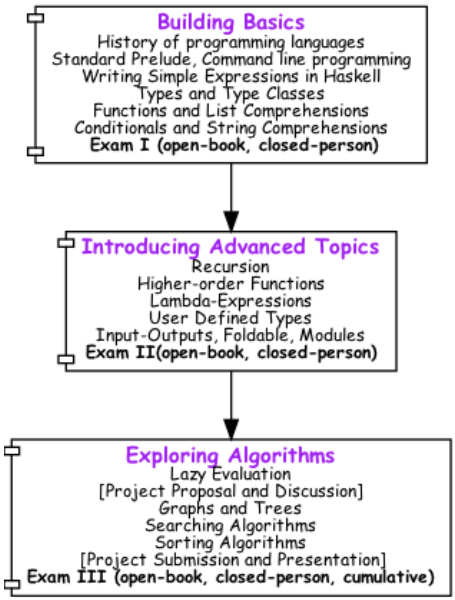


Figure 1: Topics Covered in 14-weeks

We had a 90 minutes class and a 90 minutes lab every week; the lab worked as a platform for practicing what we learned in lecture sessions. To fit the packed schedule, we sometimes overtook some lab hours for discussions and sometimes gave exercises on multiple topics in one lab. Each lecture had two 10-minute group discussion problems. Budgeting the preparation time ahead helped us in running this course smoothly. We spent roughly 14 hours per week on the course-preparation (lecture design, discussion problems, lab designing, grading, etc.).

3 Mechanics and Teaching Challenges

We started every class with this slogan: *"complete the reading before class, submit the labs on time (even partially complete), reach out for help if you are stuck, maximize the discussion time."* since we believe that computing professionals are required to leverage self-direction in their life-long learning to adapt to new emerging technologies [1]. In the next set of sections we demonstrate how we addressed some skills through different activities.

3.1 Embedding Inquiry and Analysis with Problem Solving

Out of several effective techniques for teaching programming, we followed the approach called PRIMM [7] (Predict, Run, Investigate, Modify, Make) for the labs. Figure 2 shows the steps and addressed objectives at each stage in the labs.

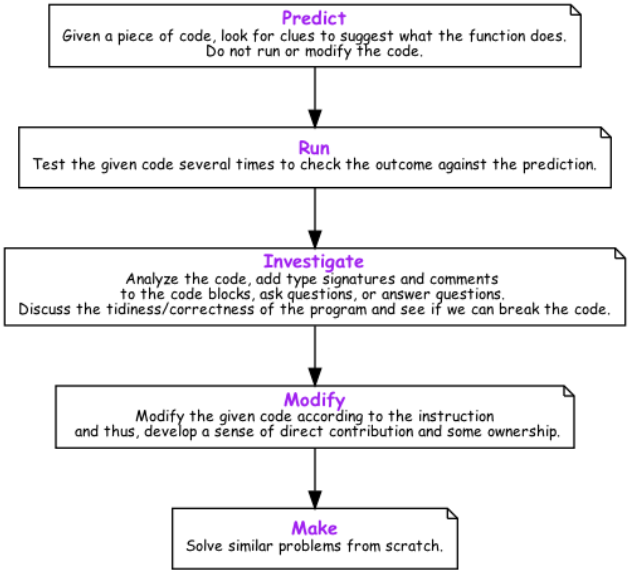


Figure 2: PRIMM approach for running the labs

Appendix A shows one sample lab problem. Considering task 1 through task 5 (each task addressing one phase from PRIMM) as a complete problem set, we wrote two to three such problem sets for each lab. While writing the lab exercises, we were mindful of the student's time, and, during lab hours, we put them in small groups of two/three to work together. We hopped into each group's discussion room from time to time to see how they approached the solutions. Had there been any joint issues, we recalled them to the lab-room and cleared the confusions.

3.2 Enhancing Critical Thinking, Teamwork, and Oral Communications

A great programmer (or a problem solver) is not necessarily always an effective communicator even though communication is one of the most crucial keys to

success. So, instead of only focusing on the syntax of the language and the functional problem solving styles, we incorporated ways to enhance critical thinking and enforce oral communication in every class. During each lecture, once we made enough progress on a new idea/topic, we assigned discussion based questions and divided students into different teams playing different roles. Students received credits for their responses by participating in two discussion questions per class. Table 1 shows a sample discussion problem.

Table 1: Sample Discussion Problem

- Given the following problem, **team A will share their solution and team B will generate/provide test cases.** Both team should consider boundary cases carefully.
- Team C will verify if the provided solution(s) passes all the test cases.
- If needed, Team A and B can switch their roles. If the solutions fail any test case or any test case is invalid, corresponding team will explain the issue and try to fix it. Team C may help both Team A and Team B.

Sample problem Function 'gcd' takes 2 integers as input and finds the greatest common divisors between them.

1. Write an expression that finds the $gcd(a_i, b_i), \forall_{i=[1...n]}\{a_i \in A, b_i \in B\}$ where A and B are two lists, each with n integers.
2. Use the following algorithm to define another function (let's call it gcd_{new}) that does the same work as gcd.

   $gcd_{new}(a, b)$:

   if $b = 0$ **then**
   return a
   else
   return $gcd_{new}(b, a \bmod b)$
   end if
3. Write an expression that finds the $gcd_{new}(a_i, b_i), \forall_{i=[1...n]}\{a_i \in A, b_i \in B\}$.
4. Write an expression that finds the least common multiple of two numbers by using the gcd (or gcd_{new}) function.

3.3 Applying Creative Thinking in Leading to a Purposeful Project

The final class project was one of the last activities to practice the learned skills thoroughly, especially team work and defining a purposeful project and make it work up to a certain standard. A successful project is the outcome of many qualities and efforts: students' personal goals and expectations, group work, professor's expectations, and demands. Students completed the following significant activities for the project:

- Step 1: Writing a proposal
  - Forming Groups of 2/3
  - Brain-storming ideas

  - Sharing thoughts with the faculty and finalizing the goals of the project
  - Writing down the project proposal following the given template
- Step 2: Implementation
  - Creating a weekly plan to save a solid amount of time for the project
  - Meeting with project partners, discussing the next steps, and distributing the tasks among the group members
  - Following the plan and completing the individually assigned tasks
  - If any issue arises, reaching out for help
- Step 3: Creating the Presentation
  - Developing some test cases
  - Creating the presentation (slides/audio, video, images)
  - Writing the project report
  - Practicing the talk before formal/final presentation
- Step 4: Presenting the final work in front of the class

Table 2: Some Projects Developed by Students

Category	Dataset analysis	Game design	Algorithmic Problem Solving
Student projects	Stroke prediction	Guess the word	Graph-based greedy algorithms
	Water quality detection	2-player battle-game	
	Mushroom identification (edible or not),		Video-game recommendations based on user preferences
	Breast cancer prediction		Automatic farm drop rates for Minecraft

We provided them a proposal template (Appendix B) and a model project (designed and solved by ourselves in seven days); the model project (a simple 2-player game) helped students understand the baseline expectations and was used as a clear standard to compare their work. Table 2 lists some students-led projects. The learning outcomes of the project were the following:

1. Define a clear goal-based problem
2. Create new user-defined data types (if required)
3. Read from and write to different types of files
4. Identify the areas that need more attention (future directions)
5. Use some new Haskell packages that were not introduced in the class
6. Find and relate old lab exercises to their projects
7. Present clearly and concisely

Students were not very clear about how to -

- predict/plan the timeline of the project ahead of time,
- define the scope of the project at the beginning.

One weakness (concern) of our policy was that majority of the students followed the style of the faculty shown for project development and presentations and relied less on their creativities. Overall, students reported that they roughly spent 20 to 27 hours completing the project.

3.4 Building Accountability, Stress and Time Management Habits

We gave three take-home exams with three to four problem sets per exam and students had five working days to complete each exam without any late submissions (unless there is a medical emergency). During exams, students got the chance to practice the sense of accountability, stress and time management as it was a multi-day exam, and they were solving the problems all on their own, allowed to discuss the issues with no one other than the instructor.

4 Student Evaluations

Students unanimously appreciated learning many new ideas and adjusting well to the FP programming style. A small group of the students found some instructions ambiguous or not specific at the beginning. They resolved the issues by coming to the dedicated office hours. Another striking comment was that they liked the project and wanted to solve more similar problems. We took it as a very positive note and may add one/two mini-project(s) by replacing some labs in the future. Due to page limit constraints, we are unable to add the comments here.

5 Self-reflection and Conclusion

Ideally, we wanted to practice peer programming in labs, which we couldn't due to the remote model, so we ended up applying team-based programming. We divided them into small groups ($2 \sim 3$ students per group) for working together in each lab/discussion. We assigned at least two discussion problems in each class (approximately 20 discussion problems in 14 weeks). The labs and discussions were primarily guided works, but students could apply the integrative knowledge and creativity in the class projects. Through the class project, we made them practice formal presentation and essay writing skills. Had it been an in-class experience, we would have practiced leadership-building skills by letting students discuss their solutions/ideas on the board regularly.

Our takeaways from teaching this course are: a) set the standards at the beginning, b) have a clear template for labs, c) make a strategic plan for the semester and break it down to weekly tasks, d) save a modest amount of time per week for the course (lecture prep, lab design, grading, helping students), e) start writing the labs and exams early, f) provide a model project; it will be a guide for the novices to set their standards, g) write moderate exams, keep the students' expertise and expectations in mind, and provide hints for each exam question. i) be mindful of the students' time and do not write lengthy labs/exercises.

Finally, the audience being characterized and the topics selected based on the demands, any programming course may follow a similar structure (PRIMM-style labs, multiple group discussions per class, team projects, and take-home exams) to practice the pedagogical elements we mentioned.

References

[1] Naomi R Boyer, Sara Langevin, and Alessio Gaspar. Self direction & constructivism in programming education. In *Proceedings of the 9th ACM SIGITE Conference on Information Technology Education*, pages 89–94, 2008.

[2] Center for Teaching Innovation. Active learning. `https://teaching.cornell.edu/teaching-resources/engaging-students/active-learning`.

[3] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, New York, NY, USA, 2013.

[4] Miran Lipovaca. *Learn You a Haskell for Great Good! A Beginner's Guide*. No Starch Press, USA, 1st edition, 2011.

[5] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. *Working group reports on ITiCSE on Innovation and Technology in Computer Science Education*, pages 204–223, 2007.

[6] Christopher Piech, Ali Malik, Kylie Jue, and Mehran Sahami. Code in place: Online section leading for scalable human-centered learning. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 973–979, 2021.

[7] Sue Sentance and Jane Waite. Primm: Exploring pedagogical approaches for teaching text-based programming in school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 113–114, New York, NY, USA, 2017. Association for Computing Machinery.

Appendices

A A Sample Lab Problem

```
{- Lab 07: Recursion and Higher Order Function
Solved by: <your name goes here> -}
{- "stage: predict"
task 1a: predict what function01 does and write a brief answer. [2
    points]
answer 1a: <your answer goes here>
task 1b: predict what data types function01 can deal with and write a
    brief answer. [2 points]
answer 1b: <your answer goes here>-}
function01 [] _ = 0
function01 (x:xs) t
        | (t == x) = 1 + function01 xs t
        | otherwise = function01 xs t
{- stage: "run"
task 2: run the code for at least three times with different inputs and
    record your test runs with corresponding outputs. [3 points]
answer 2: <your answer goes here>-}
{- stage: "investigate"
task 3: test function01 with ["goal", "go", "Go", "going", "got"] as
    the first parameter and "go" as the second parameter. record the
    test output, and explain how the output is computed by the
    function01.[2 points]
answer 3: <your answer goes here>-}
{- stage: "modify"
task 4:
Say, function01 does X operation.
a) Modify function01 such that it only works for integers.
b) Write an expression that uses function01 to do X operation on a
    particular value (say 3) from a list of lists.
c) Write an expression that uses function01 to do X operation on a
    particular list (say [1, 1, 1]) from a list of lists.
Add/modify new pieces of code in the given source file. [3 points] -}
```

```
-- Note: This is an example of blending recursion with higher-order
    functions, lambda expressions, and type declaration.
{- stage: "make"
task 5: [9 points]
a) Write a new function (call it `countOdds`) that takes a list of
    integers as input and returns the total number of odd values found
    in the input list.
b) Add at least three test cases (as multiline comments after the
    function definition).
c) Write an expression that uses countOdds to count the frequency of
    odd values from different lists of integers (a list of lists).
Add/modify new pieces of code in the given source file.-}
```

B Project Proposal Template

Project Title: Choose an informative title for the project.
Project Description: Describe the core idea of the project. State some reasons why the project idea sounded interesting.
Goals:
State a few mandatory and a few extended goals.
Development path: Split the development process of the project into several stages, and assign at least two activities for each stage. If possible, plan for parallel phases and define which partner will work on which phase. State some troubleshooting cases.
Dataset: If the project needs any publicly available dataset, introduce the dataset (size, features, purpose, link) briefly.
Timeline: Create a detailed (tentative) daily hour-based plan that your group will follow for completing the project.
Expected output State/show some manually crafted input and output of the work.
Future Directions: Expect to face some challenges and prepare to handle them. State some solid plans to extend the project given time, effort, resources.

Web Accessibility: An Evaluation of CCSC Central Plains Participants' University Home Pages*

Michael Whitney, Stephen Dannelly
Department of Computer Science
Winthrop University
Rock Hill, SC 29733
Awhitneym@winthrop.edu, dannellys@winthrop.edu

Abstract

COVID-19 required students, faculty, and staff to move quickly into the digital world. While the legal requirements for equal access to the digital world have existed for years, the rapid transition might have introduced some unexpected barriers. For persons with disabilities, these barriers can be the deciding factor as to whether or not participation occurs. Like many universities, we offer a degree program that concentrates on web development. One of the many topics that students learn is how to develop websites that conform to international standards for accessibility. But do our universities practice what we teach? This paper presents an analysis of how well the home pages of universities that participated in the 2019, 2020, and 2021 CCSC Central Plains Conference comply with accessibility standards.

1 Background

Universities' attention to web accessibility standards serves two purposes. First, our graduating students need to be proficient in the practices and procedures needed to develop and maintain online resources that align with industry standards and abide by government regulations for online accessibility. Second, it

is important that all abled and disabled students have equal access to online college and university offerings. Because of the 2017 changes to legislation, reviewed below, and the rapid expansion into the digital world due to COVID-19, we have modified how we teach digital accessibility. During this time, we became interested in how institutions are complying with the new standards during COVID. We intend to share with our fellow professors an overview of the web accessibility standards and examine how well our institutions are complying with these standards.

Congress enacted Section 504 of the 1973 Rehabilitation Act [10] and Title II of the Americans with Disabilities Act [8] to ensure persons with disabilities have equal opportunities to participate in the post-secondary education experience. Because both statutes pre-dated our digital world, Congress moved to address digital accessibility barriers by passing the Section 508 Rehabilitation Act Amendment in 1998 [11] for which web accessibility standards were enacted by the U.S. Access Board in 2000 [5]. Section 508's web accessibility standards consisted of 16 guidelines. These guidelines included guidelines for issues such as captions for videos, descriptions for pictures, and avoiding the use of seizure-inducing flicker elements. As there are only 16 accessibility areas to address, compliance does not necessarily constitute functionality for a person with a disability.

During the same time, the World Wide Web Consortium created the Web Content Accessibility Guidelines 1.0 which were then updated in 2008 (WCAG 2.0). These guidelines go into much greater depth than Section 508. With a focus on usability, WCAG 2.0 [7] organizes its guidelines into four principles of accessibility: perceivable, operable, understandable, and robust. For example, alternative text makes an image perceivable to those who are blind. An example of operability might be providing multiple means to activate buttons. To expand, if a button could only be activated with a click, then the button would not be operable to persons who do not use a mouse such as someone who is blind. The last two guideline categories are related to understandable and robust. Directions on how to use controls make them understandable and an interface is robust when it works on many devices. The 61 WCAG 2.0 guidelines include 3 levels of compliance: A (lowest - 25 guidelines), AA (13 guidelines), and AAA (highest - 23 guidelines). Each level requires conformance to the previous level.

In January 2017, the U.S. Access Board [4] modified Section 508 by replacing the previous guidelines with WCAG 2.0 Level A and Level AA. By doing so, U.S. digital accessibility standards became better aligned with international standards. Concerning WCAG 2.0, the World Wide Web Consortium continues to evolve WCAG and at the time of this research, newer versions were available. However, WCAG 2.0 is the standard supported by the U.S. Access

Board at this time.

Because the U.S. has required WCAG 2.0 for a few years, educators should (if they have not already done so) examine their instructional practice and modify/update as needed. Appropriately, this would include an update on assessment tools and their usage, accessible design practices, and usability testing. In addition, universities should review their policies and practices to stay in compliance. To this end, understanding the current state of university web site accessibility will assist both instructors and administrators.

2 Methodology

Our sample of college and university web sites is based on participants in the 2019, 2020, and 2021 CCSC Central conference. The conference participants included the Regional Board, the Conference Steering Committee, reviewers, and authors from 19 colleges and universities in the Iowa, Kansas, Nebraska, and Missouri region. We examined the home pages of those 19 institutions using three accessibility evaluation tools.

To assist developers in complying with standards, W3C provides a list of 159 Web accessibility evaluation tools that "help you determine if web content meets accessibility guidelines" [6]. Some assessment tools are free to use, while some charge a fee. Some require registration and some do not. Different tools check a variety of different standards, from US government Section 508 compliance to German government requirements. The majority of the tools check for WCAG 2.0 compliance.

Automated tools are not capable of checking all compliance requirements. Therefore, a human must manually ascertain the accessibility of multiple elements. For example, WCAG 2.0 standard 1.2.2 (level A) specifies audio and video media should have captions. That check requires a human to watch the video or listen to the audio to determine if captions exist and match the audio. There are more manual checks in WCAG 2.0 than in the first version of Section 508. Thus, institutions, instructors, and students should plan to spend more time manually checking their web pages.

Our criteria when selecting an automated tool from W3C's list included free use, easy to use, check against the WCAG 2.0 standard, provide a breakdown report of significant accessibility issues, map issues to the WCAG 2.0 standard, provide an accessibility score, and require no registration. We found no single tool provided us with everything we sought.

We used the AInspector WCAG Firefox Browser add-on from the University of Illinois at Urbana-Champaign [9] to assess websites against a set of WCAG 2.0 accessibility tests. This add-on is based on the Functional Accessibility Evaluator (FAE) 2.2 [2] which also analyzes web pages for requirements defined

by the W3C Web Content Accessibility Guidelines 2.0 Level A and AA Success Criteria [7]. Once installed, the AInspector runs alongside the visited web page and provides a summary report that allows users to explore violations and the recommended resolutions. Results are categorized as "Violations", "Warnings", "Manual Checks", and "Passed".

For a second opinion, we used A-Checker [1]. We found A-Checker's output to be user friendly, particularly in identifying specific standards that were not met. At achecker.ca/checker/, the user enters a URL to analyze, and the system provides separate tabs for "Known Problems", "Likely Problems", and "Potential Problems". We based our results on the types of Known Problems for WCAG 2.0 levels A and AA.

For a third opinion, we used TAW [3], from the Information and Communication Technology Centre Foundation (CTIC) in Spain, also applies a set of WCAG 2.0 accessibility tests to websites. At the page https://www.tawdis.net the user enters the URL of the page to be checked. Results are categorized as "Problems" (corrections are needed), "Warnings" (a human review is necessary), and "Not Viewed" (Full manual review). We based our results on TAW "Problems".

3 Scoring

Our evaluation looked at three factors. First, we used AInspector to determine how many WCAG 2.0 Level AA areas a website failed to meet. Second, we used AInspector results to build a score based on the total amount of automated results that were found to either be "Pass" or "Violation" (fail). The score is calculated as follows: pass / (pass+violation) * 100. For example: passing 59 tests and failing 16 tests = 59 / (59+16)*100 = 78.66%. This score would include multiple violations in the same category (e.g., 12 different pictures without alt tags counted as one violation).

These scores are a bit deceiving as the scoring system does not include manual checks, some involving human judgement. For example, a human is required to determine if an image needs a long description (typically no). In addition, failing an automated check might not constitute the loss of page functionality. For example, an image that does not express any meaning, such as a decorative list marker, might not have an alt tag (in this case the alt tag should be alt="). Users with visual impairments can still interact with that page.

We tested AInspector, A-Checker, and TAW on an old web page built by paper author Anonymous that was never written without any consideration to accessibility. Our score using AInspector data for the course web page was 30.76%. The page failed 9 tests, had 2 warnings, needed 20 manual checks, and

passed 4 tests. One test found multiple instances of outdated HTML tags e.g., using <b> for bolding instead of . AInspector found the course page violated multiple WCAG 2.0 standards. Some of which include: missing alt text for an image (1.1.1 - A), a navy blue text on a tan background (standard 1.4.3 - level AA), incorrectly nested section headings (3.2.3 - AA), and the page does not indicate its language (3.1.1 - A) to name a few.

AInspector, A-Checker, and TAW do not always agree on what constitutes a problem. Take the following HTML as an example (non-nested heading):

```
<body>
  <h4>Non-nested heading </h4>
```

A-Checker is okay with that heading. But AInspector reports:

Violation - 1.3.1 - Adjust the level of the h4 element or other heading elements so that the headings are properly nested on the page.

TAW reports:

Problem – 1.3.1 - No h1 element in the document

AInspector and TAW are more strictly interpreting Adaptable Guideline 1.3.1 - "Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)". TAW indicates the need for an h1 but this would not solve the nesting problem. AInspector's recommendation is more specific and the problem would be resolved if the recommendation was applied.

4 Results

Table 1 shows the results from analyzing the home pages of the 19 colleges and universities in our testbed. The table lists the name of each institution, AInspector results, A-Checker results, and TAW results.

Manual checks are part of every accessibility evaluation process. As indicated previously, the new guidelines are more comprehensive and thusly require more manual checking. To this end, Table 2 contains an average of identified manual assessments needed for WCAG 2.0 level AA as identified by AInspector, A-Checker, and TAW. The average of manual checks was based on the number of "Warnings" and "Manual Checks" for all schools. These averages do vary because each tool counts manual checks differently. On one end, AInspector groups similar checks into a single manual check, and on the other end, A-Checker lists all the checks individually. TAW does a little of both.

Table 1 shows the results from analyzing the home pages of the 19 colleges and universities in our testbed. The table lists the name of each institution, AInspector results, A-Checker results, and TAW results.

Manual checks are part of every accessibility evaluation process. As indicated previously, the new guidelines are more comprehensive and thusly require more manual checking. To this end, Table 2 contains an average of identified manual assessments needed for WCAG 2.0 level AA as identified by AInspector, A-Checker, and TAW. The average of manual checks was based on the number of "Warnings" and "Manual Checks" for all schools. These averages do vary because each tool counts manual checks differently. On one end, AInspector groups similar checks into a single manual check, and on the other end, A-Checker lists all the checks individually. TAW does a little of both.

Table 1: 2021 CCSC Central Plains Website Assessment Results

	AInspector			A-Checker	TAW
	Pass	Fail	Score	Fail	Fail
Drury University	20	9	68.9%	49	26
Fayetteville State University	24	11	68.5%	3	53
Fort Hays State University	28	6	82.3%	2	26
Grinnell College	33	6	84.3%	8	15
Kansas State Polytechnic	26	5	83.8%	8	25
Lincoln University	18	7	72.0%	48	48
Missouri State University	30	4	88.2%	1	26
Missouri Western State University	29	5	85.2%	16	26
Northwest Missouri State University	28	6	82.3%	9	6
Park University	20	6	76.9%	0	32
Saint Louis University	26	3	89.6%	0	32
St. Charles Community College	7	5	58.3%	0	16
Truman State University	16	8	66.6%	28	24
University of Missouri – St. Louis	25	11	69.4%	46	228
University of Missouri Kansas City	26	5	83.8%	6	32
University of Nebraska-Lincoln	26	6	81.2%	1	26
Washburn University	33	5	86.8%	4	33
Washington University	26	2	92.8%	2	25

Table 2: Average of Manual Checks and Warnings based WCAG 2.0

AInspector Warnings and Manual Checks	TAW Warnings and Manual Review	A-Checker Likely Problems and Potential Problems
16.53	55.26	142.08

5 Discussion

Of the home pages we evaluated, many schools obtained scores above 80%. Of this group, the top four were Washington University, St. Louis University, Missouri State University, and Washburn University. While their pages contained violations, many of the violations can be easily remedied. For example, using the HTML tag <b> for bold instead of the more recent tag . Further inspection of home pages with lower AInspector scores and higher numbers of failed WCAG 2.0 standards revealed a few significant accessibility problems. Examples include images without alt tags, forms without labels, and buttons that require the use of a mouse, therefore excluding persons with visual disabilities from fully interacting with the home page.

When assessing and updating a page for accessibility, manual checks do consume time but reveal barriers that were not found with an automated tool. As shown in Table 2, the average number of warnings and potential problems varies depending on the tool. As mentioned previously, this can be deceiving as one tool lumps issues into a single category while another tool counts all of the issues. Upon closer examination, the total amount of potential issues found is pretty close across the board. When considering the amount of time needed to perform the manual checks, many of the checks can be completed in a quick fashion (e.g. image might contain text that is not in alt text or link may not be meaningful), but the time needed to assess a webpage is going to be increased.

We also found differences in the way different tools judge accessibility. While we focused on three tools, other tools gave slightly different results. The WCAG 2.0 standard leaves room for interpretation. It is easy to imagine students or accessibility officials at any institution testing a variety of tools to find the tool that gives their work the best review.

Any instructor that has used a rubric knows that sometimes the rubric fails. Occasionally a project from one student might check more boxes on the rubric than a more impressive project from another student. Does having a simpler, more generic home page help a university's accessibility score for their homepage? In short, no. In fact, the three best performers' homepages are (in the opinion of the authors) more engaging and more attractive than the three lowest-rated homepages. Accessibility does not require sacrifices in design.

Creating a score for accessibility was difficult as, admittedly, our formula does not include manual checks. As instructors, we feel compelled to provide a score to something we have assessed. When assessing any given work, instructors typically appreciate a scoring rubric. To this end, we intend on developing a rubric to help with the assessment of websites. When considering how to score pass and fail findings, we intend to weigh WCAG 2.0 Level A guidelines more than Level AA guidelines. But how to factor in manual check issues is not

yet determined. Beyond developing a rubric, we also plan to develop our assessment tool aimed at student users. Future work also includes the development of a paper describing how we integrate accessibility into the curriculum.

References

[1] A-checker web accessibility checker. https://achecker.achecks.ca/checker/index.php/. Accessed 2021-11-29.

[2] Functional accessibility evaluator (FAE) 2.2. https://fae.disability.illinois.edu/. Accessed 2021-11-29.

[3] TAW. https://www.tawdis.net/. Accessed 2021-11-29.

[4] United states access board: About the update of the section 508 standards and section 255 guidelines for information and communication technology. https://www.access-board.gov/ict.html. Accessed 2021-11-29.

[5] United states access board: Electronic and information technology accessibility standards. https://www.federalregister.gov/documents/2000/12/21/00-32017/electronic-and-information-technology-accessibility-standards. Accessed 2021-11-29.

[6] World wide web consortium: Web accessibility evaluation tools list. http://www.w3.org/WAI/ER/tools/. Accessed 2021-11-29.

[7] World wide web consortium: Web content accessibility guidelines (wcag) 2.0. https://www.w3.org/TR/WCAG20/. Accessed 2021-11-29.

[8] Americans with Disabilities Act of 1990. 42 U.S.C. § 12101 et seq. (2009).

[9] J. Gunderson and N. Hoyt. AInspector WCAG. https://addons.mozilla.org/en-US/firefox/addon/ainspector-wcag/. Accessed 2021-11-29.

[10] Section 504 of the Rehabilitation Act of 1973. 29 U.S.C. § 794 (2009).

[11] Section 508 of the Rehabilitation Act of 1973. 29 U.S.C. § 794 (2009).

A Snapshot of Current and Trending Practices in Mobile Application Development*

Michael P. Rogers[1], *Jonathan Gratch*[2]
[1]*Computer Science Department*
University of Wisconsin Oshkosh
Oshkosh, WI 54901
mprogers@mac.com
[2]*Computer Science and Informatics*
Texas Woman's University
Denton, TX 76204
jgratch@twu.edu

Abstract

Mobile application development is a rapidly and continually evolving field. There have been dramatic changes, in some case complete paradigm shifts, in the underlying technologies. But what about mobile application development courses? Have they kept pace with those changes? In the summer of 2021, mobile application development instructors were surveyed to examine what content is being taught, what technologies are being used, and what pedagogies have been selected. This paper provides a description of how mobile application development technologies have evolved since the last major survey on the subject nearly a decade ago, then describes the findings of our survey. Results of this research will be of particular benefit to instructors who are a) tasked with designing a course on mobile application development; b) currently teaching a course and interested in ensuring that their curricula remain relevant/appropriate, and c) making an argument for resources to improve the classroom experience.

1 Introduction

It is no exaggeration to say that mobile devices have become ubiquitous in recent years, with the number of smartphones worldwide surpassing 6 billion and predicted to increase by several hundred million more in the next few years [9]. This growth has been fueled by the apps on those devices, and in industry there has consequently been a commensurate rise in the demand for sophisticated, polished mobile apps that can be rapidly developed and modified. Meeting this demand has become possible largely due to dramatic changes, in some case complete paradigm shifts, in the underlying technologies used to develop apps. But what about mobile application development courses? Have they kept up with these changes, i.e., are we meeting the needs of our students' future employers, and how is the pedagogy evolving? There are no good recent surveys to address these questions, the answers to which are absolutely vital for those teaching the subject. Faced with these questions, we created a new survey aimed at capturing a snapshot of the current curricular, technological and pedagogical practices in use within mobile app development classes.

1.1 Terminology

Throughout this paper, we follow convention and broadly categorize mobile apps as native, cross-platform, or web-based. *Native apps* are developed using the Software Development Kit (SDK) and tools provided by the maker of the operating system, and the binaries that result run only on that specific platform. *Cross-platform apps*, in contrast, use a single codebase to create binaries that will run, and provide the same user experience, on multiple platforms. Finally, *web-based apps* run within the confines of a web browser: because their development is quite distinct, however, and is typically taught in a separate course, they were omitted from the survey.

1.2 Background

One way to grasp the Darwinian nature of the mobile landscape is to look back at a comprehensive review of mobile application development and its integration into computer science courses [4] in 2012. Then, native apps could be developed for iOS, Android, Windows Phone and Blackberry (the latter two have been discontinued). SDKs for both iOS and Android were available, based on the programming languages Objective-C and Java (the SDKs have grown considerably, and their respective preferred development languages are now Swift and Kotlin). In 2012, cross-platform apps could be created using Apache Cordova (PhoneGap) and TouchDevelop (the former was abandoned by its original champion, Adobe [1], although it lives on as an open source

project; the latter is defunct). Web apps, common in 2012, continue to be so today.

Burd et. al. [4] further found that in 2012, mobile computing had been incorporated into computer science programs in many ways - from inclusion as units within courses all the way to standalone courses. One of the primary challenges in standalone mobile app development courses was that they were typically upper level courses and were introduced late in a student's course of study [10, 6]. Their late appearance in the curricular sequence in many programs was indicative of the constant change in content and requirements that students be knowledgeable in many areas, including database, networking, and HCI concepts. The timing of the course, the choice of platform(s) and the content selected for the course (e.g., native versus cross-platform) all presented challenges for instructors. The importance of all these considerations was highlighted by the creation of a separate knowledge area on Platform-Based Computing in the ACM/IEEE Curriculum 2013 report [11].

Since 2012, we have witnessed a proliferation and maturation of cross-platform tools used in mobile app development [3]. Flutter (from Google, programmed in Dart), Xamarin Forms (Microsoft, C#), React Native (Facebook, JavaScript) and NativeScript (OpenJS Foundation, JavaScript) are the most popular general-purpose examples [7], but the two most prevalent major game engines (Unity, Unreal Engine) are also cross platform.

Previous common objections to cross-platform SDKs, primarily a) their inability to access on- device sensors and databases, and b) deviations from the canonical user interface standard produced by the native SDKS, have been largely dealt with. Modern cross-platform apps can access sensors (GPS, accelerometer, compass, gyroscope, camera) and tap into the encrypted keychain/keystore services provided on their respective systems. In addition, cross-platform apps can be developed that are visually and functionally virtually indistinguishable from their native counterparts.

1.3 User Interface Paradigms

One of the most challenging aspects of app development is managing and maintaining an app's data or state. Native apps have traditionally been based on imperative frameworks, meaning that when the state of the app changes, the developer is responsible for explicitly changing the User Interface (UI) to reflect that; and conversely, if a user manipulates a control, the developer must then use its value to update the state. Implementing the methods to accomplish this is relatively straightforward, and students grasp the concepts readily. However, as apps increase in size, keeping the UI in sync with its state becomes increasingly problematic.

The cross-platform SDKs have championed an alternative declarative ap-

proach. In this style of app development, the developer creates the UI, and declares in code how the UI should look for a given value of state. When the state changes, the UI is updated automatically. This eliminates an entire category of bugs in which the UI fails to properly reflect state (or conversely, when the UI is manipulated, state is not properly updated). This approach has turned out to be so successful that both of the major native SDKs are now transitioning towards this declarative UI style: Apple's SwiftUI and Google's Jetpack Compose.

2 Survey Results

In the summer of 2021, we sent out a call to various mailing lists (including SIGCSE, CCSC, CSTA, and EDUTECH), asking for instructors teaching courses in mobile application development to complete a survey. We offered one minor enticement, the option of being entered into a drawing for an Amazon Gift Card, and one major one: the ability to contribute to our understanding of the current state of mobile application development courses. We did not specify a deadline. The survey covered course structure, content, tools, hardware, and pedagogy, in a survey instrument consisting of 67 multiple choice and multiple answer questions. The results were analyzed using SAS.

A total of 90 people responded to our request. Of those, 81 respondents indicated that they delivered a full course (or module covering mobile application development), and went on to complete the survey, thus providing a valuable snapshot as to what is currently being taught in the classroom, and how it is being taught. Respondents were primarily from the United States, but responses were received from all over the world including Europe, Canada, and Australia. Participants were mostly of instructors at the undergraduate level, several high school and middle school educators participated. 97% of the responses indicated that the educator was faculty at that institution.

2.1 Courses Overview

Most (90.1%) of our respondents taught mobile app development as a full course; only 9.9% taught it as a component within another course. The courses were typically 3 (56.3%) or 4 (27.5%) credit hours (or the equivalent in other parts of the world); most (83.9%) were electives. They were mainly taught to undergraduates (76.5%), or undergraduates and graduates (12.3%). The majority were structured as lecture and lab (56.8%), a significant number (40.7%) were lecture only, and 2.5% were lab-only.

Nearly half (48.0%) of the courses were offered just once per year; 35.6% were offered less frequently and the remainder, 16.4%, were offered every semester. Considering the speed with which mobile app development changes,

the implication is that committing to the course is, for many, a commitment to continual course revision.

Instructors were almost exclusively full-time faculty (86.4%). The majority of classes were between 1-25 students (65.0%), with an approximately equal number of respondents teaching classes of 26-50 students (16.3%) and 50-100 students (17.5%).

Reflecting the influence of COVID-19, 63.0% of courses were exclusively face-to-face. Just under one-fifth (19.8%) were hybrid, a mixture of online and in-person meetings; and 17.3% were completely online. Most (68.1%) of the online courses were taught synchronously, with 21.3% asynchronous and the remainder a combination of the two.

At first glance this area seems as well-suited as any in computer science for hybrid or online delivery. However, as we will see in the pedagogy section, many of the courses also involved a significant final project in which students were organized in teams: and there are logistical advantages to having students and instructors all in one room.

2.2 Native App Development versus Cross Platform Development

Table 1 shows what percentage of instructors taught native app development in Android, iOS, or both, and what percentages were used by students developing apps. These numbers reflect the relative popularity of the two operating systems [8] and also the more inclusive nature of Android development, which can be conducted on Linux, macOS, and Windows computers: native iOS development requires a macOS computer. There are cloud solutions to the latter that we discuss later in this paper.

Table 1: Native App Development

Operating System	Taught/ Discussed	Developed in by Students	Worldwide Marketshare
Android	53.3%	56.4%	72.8%
iOS	13.05%	11.5%	26.3%
Both	27.3%	26.9%	-
Neither	1.3%	1.3%	-

Cross-platform app development was much less likely to be taught or discussed: only 40% of respondents indicated that they taught or discussed cross-platform in their classes. Among industry developers the number is slightly higher, approximately 50% use cross-platform technologies or frameworks, according to a survey conducted by JetBrains in 2020 [5]. Our respondents roles

as early adopters is unsurprising given the natural curiosity, the academic freedom, imperative to keep up with new trends, and the clear advantages of cross-platform in a resource-scarce environment such as academia.

Table 2 shows the most commonly taught or discussed cross-platform frameworks, sorted in decreasing order of popularity. Also mentioned were App Inventor 2, Codename One and Progressive Web Apps. It is interesting to compare these numbers to the results on the industry usage of cross-platform technologies from [5]. While only one of our respondents mentioned Ionic, among developers 18% mentioned using it. None of our respondents taught Apache Cordova, but 18% of industry developers had used it in 2020.

Table 2: Cross Platform App Development

Cross-platform Framework	Taught/ Discussed	Industry Usage
React Native	30.6%	42%
Flutter	27.8%	39%
Xamarin Forms	22.2%	14%
Native Script	16.7%	5%
Ionic	2.8%	18%
Apache Cordova	0.0%	18%

2.3 Hardware

The iOS and Android simulators included with the SDKs are convenient, enabling students to run their apps on a wide range of virtual devices. However, the interactions typically take place with a mouse or trackpad: to properly evaluate an app requires testing on physical devices. The logistics involved in ensuring that every student has access to a device are daunting and among our respondents, only 59.0% mandated their use. When asked about provided hardware (Table 3), almost half (49.3%) provided none: among those that did, more provided Android devices (the more predominant device) and macOS computers (essential for iOS development, but the less common platform among students). Another potentially less expensive way to gain access to macOS hardware is via infrastructure-as-a-service providers, such as Mac Stadium. A significant number (8.8%) of instructors indicated that they did subscribe to such a service.

Table 3: Hardware Types Supported by Institution

Hardware	Provided by Institution
Android	35.8%
iOS	9.9%
Computers (macOS)	9.9%
Computers (Windows)	4.9%
Computers (Chromebooks)	1.2%

2.4 Development Tools

Mobile app development courses can be an appropriate place in the curriculum for providing exposure to a variety of development tools. Among the reported Integrated Development Environments (IDEs), the most common were Android Studio (77.2%), Visual Studio (10.1%), Visual Studio Code (17.7%) and Xcode (31.6%).

Apart from IDEs and collaboration software, version control is one of the most important software development tools, but if not properly used can lead to disastrous results. This may explain why only 57.0% of respondents indicated that their students use Git or other version control software. With its ubiquity in industry as a fundamental tool in software development, our result suggests the need for more instructors to teach and incorporate it in course assignments.

Among other development tools, collaboration tools were used by 17.7% of students, issue management software by 6.3%; and prototyping tools by 11.4%.

2.5 Course Assignments

The majority of instructors favored project-based learning with an internal or no client (67.5%), 28.8% characterized their classes as lecture-only and a very few made use of an external client (3.8%). Smaller assignments tended to be completed individually (76.2%), with 23.8% having their students work in teams. Nearly all instructors had their students develop a significant final project (95.0%). Of those, 66.2% had students work in groups, but more than one-third (33.8%) had their students work individually.

Creating mobile apps requires in-depth knowledge of a broad range of topics, and fitting all that into a single course can be challenging. Our respondents appear to do a good job covering the mechanics: UI layout, storage (on device, and backend), and major APIs were all covered extensively (see Table 4). Topics that did not advance the immediate goal of app creation - notably data privacy, data ethics, and mobile security, received little attention. Their low representation in curriculum appear to be opportunities lost for growing

students in other aspects of software development for mobile application, especially mobile security and data privacy which are increasing concerns for developers, users, and industry alike.

Deployment to an app store can be an rewarding experience for students, and a positive way to end the semester, when their app is at last ready for use by a client or the public. However, there are obstacles to this: a student is required to pay a one-time fee for access to the store ($25 for Google Play, $99 annually for Apple's App Store), the deployment is a multi-step, somewhat arcane process, and there is an expectation of quality among the submissions that student projects may not rise to. Understandably, then, instructors in the survey either just lectured/discussed it (45.1%), or did not cover it at all (52.9%); only one respondent included it as part of a major assignment.

Table 4: Topic Coverage

Topic	Lecture/ Discussion	Major Assignment	Minor Assignment	Not covered
Storage (on device)	71.8%	46.5%	56.3%	5.6%
Testing	66.7%	22.7%	28.8%	12.1%
Accessibility	63.3%	5.0%	23.3%	25.0%
Storage (backend)	60.9%	36.6%	35.9%	23.4%
Data Privacy	50.9%	1.7%	5.1%	42.4%
Security	48.2%	3.6%	1.8%	44.6%
Audio	47.1%	3.9%	29.4%	43.1%
Networking	46.6%	27.6%	24.1%	34.5%
Drawing	46.3%	11.1%	29.6%	44.4%
Data Ethics	46.3%	1.9%	13.0%	48.2%
Deployment to app store	45.6%	3.5%	0.0%	50.9%
Video	34.7%	2.0%	10.2%	53.1%
Bluetooth	23.5%	2.0%	5.9%	68.6%
VR/AR	19.6%	0.0%	2.0%	78.4%
Machine Learning	16.4%	1.8%	7.3%	78.2%

2.6 Declarative Frameworks

In the cross-platform realm, creating declarative UIs is the norm, but the ability to do so using native SDKs is a relatively recent phenomenon. Considering the amount of time required to retool a course, it is impressive that 17.6% of instructors said that they taught SwiftUI (approximately 50% of those who said that they taught iOS). A significantly smaller fraction, 10.0%, reported

that they taught the Android equivalent, Jetpack Compose. Table 5 indicates the three primary reasons provided for avoiding SwiftUI (among those who already teach native iOS) and Jetpack Compose (among those who teach native Android). In the comments, instructors expressed concerns about the relative complexity of the topic for students, the fact that the products are beta and "too niche". The reasons are certainly valid, but the trends suggest that instructors need to be looking at this carefully.

Table 5: Reasons for not Teaching SwiftUI / Jetpack Compose

Reason for not teaching...	SwiftUI	Jetpack Compose
Unfamiliarity	48.18%	71.4%
Lack of time	29.6%	8.9%
Insufficient resources	14.8%	1.8%

3 Discussion

From our survey conducted in 2021, we found that current mobile app development courses fell heavily in line with previous research conducted in 2012, with the majority of courses being taught as an upper-level elective at the undergraduate level. The placement of these courses toward the end of many CS programs underscores the complex nature of mobile app development and the wide knowledge base needed by students. Such placement means that the educator has more technological and pedagogical choices, and therefore decisions, that must be made prior to teaching it.

The first technology decision involves the choice of mobile platform OS: Android, iOS or both. The OS-agnostic nature of the Android SDK enables development on Linux, macOS and Windows machines. In contrast, the iOS SDK requires a macOS, Xcode-equipped machine. Therefore, the financial barriers to using Android for development are generally lower than for iOS, as educators and students can utilize almost any existing computing system. Choosing iOS could require additional expenditures by the institution, or require students to acquire their own Apple Mac computers. Between this, and the fact that Android has more marketshare, it was not surprising that the educators responded with a large majority choosing the Android platform as their development platform.

When it came to the decision of teaching native or cross-platform technologies, the former was clearly the preferred choice, as native technologies are far more established, more familiar to educators, and appeal to potential industry employers who can afford two specialized development teams. However,

cross-platform is gaining popularity in industry and in the classroom, being heavily promoted by Facebook (React Native), Google (Flutter) and Microsoft (Xamarin Forms and its successor, MAUI). Also, opting for the cross-platform approach makes the first technology decision mentioned above moot: any device will do which is a huge benefit in resource-strapped academic environments. This is supported by our findings that while most of the respondents preferred the more established native approach, a healthy 40% did teach or discuss cross-platform in the classroom and represents a change from a decade ago.

Declarative frameworks are being taught by relatively few instructors, with the usual suspects - unfamiliarity, lack of time, and insufficient resources - being the major hurdles. Over time, however, the trend is clear: cross-platform frameworks rely on declarative frameworks, and the native SDKs are gravitating towards them, so that at some point in the next few years courses will have to evolve to teach this.

Accessible computing is not a new concept in computer science education [2]. With people relying more and more on their mobile devices, the need to bring awareness of accessibility in mobile app development classes is high. Our result shows that accessibility was taught or discussed by 63.3% of respondents while 25% responded with not covering this topic. We hope that the number teaching this essential topic continues to trend upwards throughout mobile application courses and CS education as whole so that we may close the gap and design and develop apps that are accessible to all.

The technology that we teach is ephemeral, but the ability to learn, and to work on teams, will always be critical to our students' success in the workforce. Our respondents understand that, as most have their students working in groups to develop a significant final project, applying at least some of the tools and practices that employers look for in graduates. Almost all have sidestepped the perils involved in client-directed projects which are appropriate in a software engineering course, but difficult to manage in a course that represents a deep-dive into a specific technology.

4 Conclusions

Overall our survey of teachers of mobile app courses suggests that instructors are teaching and students are learning most of the skills and concepts they need to know to succeed in industry. Whereas some aspects of the mobile app development course have remained the same for the past decade - particularly in the course's placement in students' program of study and the use of native technologies, educators have transitioned to newer native development technologies, such as Objective-C to Swift. Significantly more instructors are

focused on Android, in keeping with its share of the market and typically lower barrier to entry, but iOS is not being ignored. The tools that employers expect their developers to use are largely being taught, although not necessarily to the extent that they could be. The vital experience that students need, such as working in teams, is being provided in the majority. A substantial number of instructors are tackling cross-platform frameworks, and even though natively implemented declarative UIs are quite new, they too are getting significant attention in the classroom.

All of these results will offer guidance to instructors who are:

1. tasked with designing a course on mobile application development;
2. currently teaching a course and interested in ensuring that their curricula remain relevant/appropriate;
3. making an argument for resources to improve the classroom experience.

4.1 Future Work

This paper represents a snapshot of current mobile app development courses based on survey results provided by educators who are currently teaching those courses. It is an attempt to capture the curricula, pedagogical approach, and technologies used by mobile app development instructors since the 2012 survey [4]. We live in a world in which mobile is dominant, therefore it is crucial for us to continue monitoring trends in mobile application development in both the classroom and industry. To this end, we plan to periodically revise and re-administer this survey in order to meet the needs of a global audience; increase response rate through its distribution during the academic semester (rather than the summer); and reduce survey fatigue by shortening the survey and removing redundant items. To gather contextual information on course design and practices we plan to interview a select number of participants. It is our hope that instructors will take the results of this survey, compare it to what they are doing, and on that basis make adjustments, if needed, to their curricula.

5 Acknowledgements

We would like to thank the instructors who volunteered to beta test our survey, and to those who took the time to complete the final version.

References

[1] Adobe. Update for customers using PhoneGap and PhoneGap build, Aug 2021. https://blog.phonegap.com/update-for-customers-using-phonegap-and-phonegap-build-cc701c77502c.

[2] Catherine M. Baker, Yasmine N. El-Glaly, and Kristen Shinohara. A systematic analysis of accessibility in computing education research. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 107–113, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Andreas Biørn-Hansen, Tor-Morten Grønli, Gheorghita Ghinea, and Sahel Alouneh. An empirical study of cross-platform mobile development in industry. *Wireless Communications and Mobile Computing*, 2019, 2019.

[4] Barry Burd, João Paulo Barros, Chris Johnson, Stan Kurkovsky, Arnold Rosenbloom, and Nikolai Tillman. Educating for mobile computing: Addressing the new challenges. In *Proceedings of the Final Reports on Innovation and Technology in Computer Science Education 2012 Working Groups*, ITiCSE-WGR '12, page 51–63, New York, NY, USA, 2012. Association for Computing Machinery.

[5] JetBrains. The state of developer ecosystem 2020, Aug 2021. https://www.jetbrains.com/lp/devecosystem-2020.

[6] Amruth N. Kumar. Collateral learning of mobile computing: An experience report. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 2018, page 27–32, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Shanhong Liu. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021, Aug 2021. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/.

[8] S. O'Dea. Mobile operating systems' market share worldwide from January 2012 to June 2021, June 2021. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

[9] S. O'Dea. Smartphone subscriptions worldwide 2016-2021, Aug 2021. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[10] Roy P. Pargas, Punit Kulkarni, Greg Edison, and Barbara J. Speziale. Teaching mobile app software development is a challenge! (abstract only). In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 721, New York, NY, USA, 2014. Association for Computing Machinery.

[11] M Sahami and S Roach. ACM/IEEE-CS joint task force on computing curricula (2013). *Computer science curricula*, 2013.

Flutter: n Platforms, 1 Codebase, 0 Problems*

Workshop

Michael P. Rogers[1], Bill Siever[2]
[1]Computer Science Department
University of Wisconsin Oshkosh
Oshkosh, WI 54901
mprogers@mac.com
[2]McKelvey School of Engineering
Washington University in St. Louis
St. Louis, MO 63130
bsiever@wustl.edu

Flutter is an open-source, cross-platform development kit from Google that allows developers to create apps for iOS, Android, web, and desktop, from a single codebase.

Flutter takes a modern approach to app development. The SDK is based on a declarative UI paradigm that is widely used in the workforce. Apps are written in Dart, a modern, strongly typed, easy-to-learn language that reinforces good coding habits. Development can be done in Visual Studio Code, which students appreciate for its flexibility and simple interface.

Flutter allows instructors to sidestep the issue of what platform to target: students with any smart phone and operating system can now participate, sparing the university from having to provide a development environment. It provides a vehicle and means for introducing declarative UI design into the curriculum, so that when students do see this in industry, they will be prepared for it.

This technology could be used in a mobile computing class, a software engineering / capstone class, or any other situation where students need to develop for multiple platforms.

In this workshop, participants will be introduced to the language, tools, and paradigms that drive Flutter, and provided with tips, based on the presenters' experience, on how best to incorporate this SDK into the curriculum.

A Way to Visualize Higher Dimensional Arrays, Matrices, and Spaces*

Nifty Assignment

Cong-Cong Xing[1], *Jun Huang*[2]
[1]*Department of Mathematics-Computer Science*
Nicholls State University
Thibodaux, LA 70310
cong-cong.xing@nicholls.edu
[2]*Department of Computer Science*
Baylor University
Waco, TX 76798
huangj@ieee.org

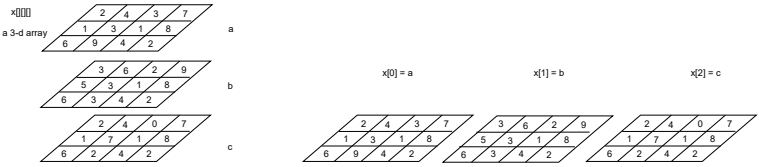
Students deal with multi-dimensional arrays/matrices/spaces (AMSs) routinely in computer science and math courses. Generally speaking, students have a clear image in their minds about lower dimensional (1-d to 3-d) AMSs, but they have trouble in visualizing what a higher dimensional (and thus abstract) AMS may look like. The purpose of this nifty assignment is to provide a way for students to "see" the higher dimensional AMSs or make these abstract entities concrete.

The connection between arrays in computer science and matrices in math is obvious: an n-d array can be equivalently viewed as an n-d matrix, which, in turn, can be viewed as sitting in an n-d space (e.g., $\mathbb{R}^n$ or $\mathbb{N}^n$). While students have no problem of picturing a lower dimensional AMS — a 1-d AMS is a line of numbers, a 2-d AMS is a table of numbers, and a 3-d AMS is a cube of numbers — they do have difficulties in doing so for a higher (> 3-d) dimensional AMS, and textbooks typically do not address this issue (as far as we know). However, being able to visualize higher dimensional AMSs is important in terms of learning and understanding anything related to these abstract entities in both computer science and math. Toward providing such a visualizing tool, we propose the following conceptual viewpoint of 2-d arrays and devise a related programming assignment in Java to let students "verify"

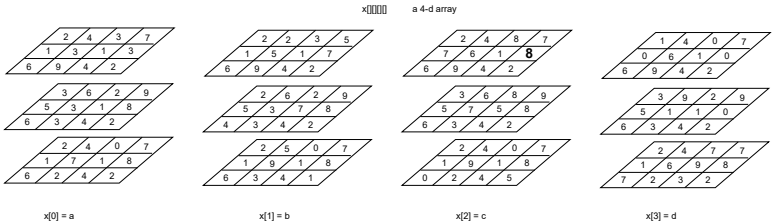
the correctness of this viewpoint by completing the assignment: *a 2-d array is a 1-d array of 1-d arrays.* For example, the following 3×4 2-d array x[][] on the left can be viewed as a 1-d array in the middle which consists of 3 elements `a,b,c`, where each of the 3 elements *itself* is a 1-d array of 4 elements as shown on the right.

```
x[][]: 3 4 5 4      x[0] = a        a = 3 4 5 4
       1 2 9 8      x[1] = b        b = 1 2 9 8
       3 1 7 6      x[2] = c        c = 3 1 7 6
```

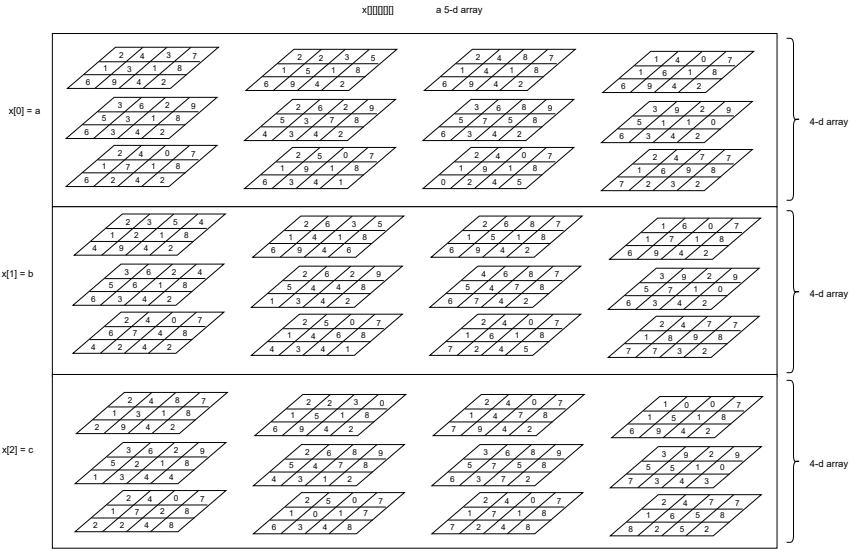
The related programming assignment asks students to pass one of the 3 elements in the middle array (say, x[1]), as an argument to a method that takes a 1-d array as parameter and and performs some actions on this array parameter (e.g., reverse the order of its elements), to see that x[1] indeed can be treated as a 1-d array. Once students are convinced that a 2-d array can be thought of as a 1-d array of 1-d arrays, this idea is extended to 3-d arrays. That is, *a 3-d array can be regarded as a 1-d array of 2-d arrays*, as shown below where the 3×3×4 3-d array x[][][] on the left can be viewed as a 1-d array of 3 elements a, b, and c, and each of them is a 3×4 2-d array.



Along the same line, this idea is expanded to higher dimensional arrays. Specifically, *a 4-d array is a 1-d array of 3-d arrays.* An example of a 4×3×3×4 4-d array is given below which shows that the 4-d array x[][][][] is a 1-d array of 4 elements a, b, c, and d, where each of them is a 3×3×4 3-d array. In particular, the element specified by x[2][0][1][3] (boldfaced in the array) can be found in the following way: the first index [2] indicates that the element being sought is in the third element x[2] of the 1-d array represented by x, the next three indexes [0][1][3] tell us how to pinpoint the element in the 3-d array represented by x[2] (which is easy to do since we are in a 3-d array now).



As the final example, a 3×4×3×3×4 5-d array is given below illustrating the notion that *a 5-d array is a 1-d array of 4-d arrays.*



This assignment was given and discussed in a freshman-level introductory Java programming class, and students' responses were basically that they had no idea about how to picture a 4-d AMS until now. Potential benefits from this assignment are summarized as follows.

- It provides a concrete approach for students to visualize and analyze abstract higher dimensional AMSs.
- Since the central idea of this approach is that an n-d AMS can be decomposed to a 1-d AMS and multiple $(n-1)$-d AMSs, its connection to the notions of recursion and mathematical induction is apparent. As such, this approach may lay some preliminary foundation for students' future studies in recursion and mathematical induction.
- Although the assignment asks students to verify that a 2-d array can be regarded as a 1-d array of 1-d arrays, it can be easily adapted to work with 3-d (or higher dimensional) arrays in the same fashion. Namely, students can be asked to decompose a 3-d array x[][][] into a 1-d array of 2-d arrays, and pass one of the x[i]'s as the argument to a method that requires a 2-d array parameter and does some operations to the parameter, to validate the correctness of this notion.

We hope that this assignment can be found useful by colleagues.

Challenges Developing and Teaching Online Professional Courses for Technical Graduate Programs *

Panel Discussion

Ajay Bandi, Denise Case, Nathan Eloe,
Aziz Fellah, Charles Hoot
Northwest Missouri State University
Maryville, MO 64468
{ajay, dcase, nathane, afellah, hoot}@nwmissouri.edu

This panel discussion will focus on experiences and recommendations from faculty designing and teaching online professional courses for a technical graduate program. Teaching in any field where content evolves quickly can be demanding. Add in the complexities of online, asynchronous instruction with remote students in highly varied environments with diverse backgrounds, and course development and production can become challenging[1, 3]. Programming languages evolve, tools get updated, online instructions change links, and providing current, concise, yet comprehensive assistance can be surprisingly difficult.

Professional students are often intelligent and successful but may have little formal training with computers[2]. Assuming a comfort level with common computer tasks may lead to surprises for the instructor. Not all students are comfortable (at first) with creating folders and files, understanding paths, or finding files stored under their username. Alternatively, some students have long since mastered these tasks and don't need remediation or foundational instructions. Multiple environments may be needed, each with slightly different installations and tools, e.g., cloud service providers, Mac, Linux, and Windows.

Recommendations from this discussion will help educators engage more effectively in this increasingly popular mode of instruction.

References

[1] Charles Badami, Ajay Bandi, Denise Case, Aziz Fellah, and Mahmoud Yousef. Engaging graduate students during the pandemic: panel discussion. *Journal of Computing Sciences in Colleges*, 36(6):78–79, 2021.

[2] Carla Brodley, Megan Barry, Aidan Connell, Catherine Gill, Ian Gorton, Benjamin Hescott, Bryan Lackaye, Cynthia LuBien, Leena Razzaq, Amit Shesh, et al. An MS in CS for non-CS majors: Moving to increase diversity of thought and demographics in CS. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1248–1254, 2020.

[3] Xi Lin and Li Gao. Students' sense of community and perspectives of taking synchronous and asynchronous online courses. *Asian Journal of Distance Education*, 15(1):169–179, 2020.

Teaching Multiple Graduate Sections with Large Class Sizes *

Panel Discussion

Ajay Bandi, Denise Case, Nathan Eloe,
Aziz Fellah, Charles Hoot
Northwest Missouri State University
Maryville, MO 64468
{ajay, dcase, nathane, afellah, hoot}@nwmissouri.edu

This panel discussion will focus on how educators manage multiple sections with large class sizes (∼60) to achieve student learning outcomes. Graduate enrollment has tripled in the fall 2021 semester in several universities within the US. It is challenging for educators to focus on every student in a large section every day. The discussion looks at various teaching practices and approaches that can help students learn programming skills in this environment. The panel also addresses the assessment and evaluation strategies for multiple large sections. We will look at how we managed the technical performance issues in the Canvas learning management system when 300 students tried to take an exam at the same time. In addition, classroom engagement was encouraged through small group discussions and collaborations using the GitHub version control system with approximately 60 pull requests per section per day. Each student provided a unique contribution to the source code where the whole class was working on a large project. While it has several challenges, students learned to communicate with others to push their code. The recommendations from this panel will help educators to teach and modify their pedagogy for large sections without compromising the necessary competencies.