The Journal of Computing Sciences in Colleges

Papers of the 29th Annual CCSC Midwestern Conference

October 7th-8th, 2022
University of Wisconsin-Stout
Menomonie, WI

Baochuan Lu, Editor
Southwest Baptist University

Saleh Alnaeli, Regional Editor
University of Wisconsin-Stout

Volume 38, Number 4 **November 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 **5**

CCSC National Partners **7**

Welcome to the 2022 CCSC Midwestern Conference **8**

Regional Committees — 2022 CCSC Midwestern Region **10**

Cutting Edge Advances in Computer Science and Our Futures — Banquet Address **12**
Kurt Heckman, Eastern Nazarene College

Building a Software Company — Keynote Speech **13**
Kurt Heckman, Eastern Nazarene College

Examine the Use of Virtual Worlds for Modeling, Prototyping, and Testing Purposes in a Classroom Setting **14**
Imad Al Saeed, Saint Xavier University

Lessons Learned Teaching Programming During the Pandemic **22**
Jean Mehta, Saint Xavier University

A Cost Estimation Model for Scrum Projects **30**
Xingzhan Feng, Kasi Periyasamy, University of Wisconsin-La Crosse La Crosse

Integrating The OpenMind Platform into a Human-Computer Interaction Elective **38**
Paul Gestwicki, Ball State University

Strategies for Equitable Participation in an Introductory Computer Science Course **48**
Meredith Moore, Timothy Urness, Drake University

Reflective Curriculum Review for Liberal Arts Computing Programs — Conference Tutorial **58**
Jakob Barnard, University of Jamestown, Grant Braught, Dickinson College, Janet Davis, Whitman College, Amanda Holland-Minkley, Washington Jefferson College, David Reed, Creighton University, Karl Schmitt,

Trinity Christian College, Andrea Tartaro, Furman University, James Teresco, Siena College

Interdisciplinary Project-Driven Learning in Game Design and Development — Panel Discussion **61**
Seth Berrier, Karl Koehle, Kimberly Long Loken, Michael Tetzlaff, Tyler Thomas, University of Wisconsin – Stout

Flutter: n Platforms, 1 Codebase, 0 Problems — Conference Workshop **67**
Michael P. Rogers, University of Wisconsin Oshkosh, Bill Siever, Washington University in St. Louis

Summary Words and in the News — Nifty Assignment **69**
David L. Largent, Ball State University

Computer Network Between Two Departments Using Cisco Packet Tracer — Nifty Assignment **70**
Imad Al Saeed, Saint Xavier University

Computer Disassemble and Rebuild Days — Nifty Assignment **71**
James Roll, University of Findlay

Buried Wireless Sensor Node Based on Internet Wi-Fi and Bluetooth technology for Precision Agriculture — Work In Progress **72**
Ahmed A. Elmagrous, University of Wisconsin-Stout Menomonie

Encouraging Student Voice with D, E, I Based Online Communication Standards — Work In Progress **74**
Kristi Hall, University of Cincinnati Batavia

Teaching Programming Paradigms Using CLIPS — Work In Progress **78**
Ramachandra B. Abhyankar, Indiana State University Terre Haute

Attendance Mobile Application — Work In Progress **79**
Zoltan Nahoczki, Matthew Fallon, Reed Mitchell, University of Wisconsin Parkside Kenosha

Reviewers — 2022 CCSC Midwestern Conference **81**

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:

Chris Healy, President (2024), chris.healy@furman.edu, Computer Science Department, 3300 Poinsett Highway Greenville, SC 29613.

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

Brian Hare, Treasurer (2023), 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 (2025), cathy.bareiss@betheluniversity.edu, Department of Mathematical Engineering Sciences, 1001 Bethel Circle, Mishawaka, IN 46545.

Judy Mullins, Central Plains Representative (2023), Associate Treasurer, 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(2025), dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg, IN 46112.

David Largent, Midwest Representative(2023), dllargent@bsu.edu, Department of Computer Science, 2000 W. University Avenue Muncie, IN 47306.

Mark Bailey, Northeastern Representative (2025), mbailey@hamilton.edu, Computer Science Department, 198 College Hill Road, Clinton, NY 13323.

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

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

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

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

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

Serving the CCSC: These members are serving in positions as indicated:
Bin Peng, Associate Editor, bin.peng@park.edu, Park University - Department of Computer Science and Information Systems, 8700 NW River Park Drive, Parkville, MO 64152.
Ed Lindoo, Associate Treasurer & UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, 3333 Regis Boulevard, Denver, CO 80221.
George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood college, 401 Rosemont Ave. Frederick, MD 21701.
Megan Thomas, Membership System Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, One University Circle, Turlock, CA 95382.
Karina Assiter, National Partners Chair, karinaassiter@landmark.edu.
Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

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

Platinum Level Partner

Google Cloud
GitHub
NSF – National Science Foundation

Gold Level Partner

zyBooks
Rephactor

Associate Level Partners

Mercury Learning and Information
Mercy College

Welcome to the 2022 CCSC Midwestern Conference

On behalf of the conference committee, I welcome you to the 29th annual Consortium for Computer Science Conference (CCSC) for the Midwest region. We are very excited about hosting our conference this year at the University of Wisconsin-Stout in Menomonie, Wisconsin.

The conference program includes refereed paper sessions, panels, speakers, workshops, tutorials, nifty assignments, student activities, and vendor sessions. The papers that were submitted were of high quality and the acceptance rate was 60%. All accepted papers will be presented and published in the journal of Computing Sciences in Colleges. Sessions cover a variety of topics and research areas including computer science education, smart farming, precision agriculture, human-computer interaction, game design and development, sensor networks, modeling and software testing, and mobile applications. We will also be hosting our known programming contest for our undergraduate students as well as different activities focused on students including the Student Showcase, and a popular panel session titled "What Students Need to Know About Industry". We have representatives from seven different companies as part of this panel.

Two of our National Partners, Google, and zyBooks, are hosting sessions. I sincerely thank them for their support of CCSC as National Partners. I encourage you to meet with their company representatives and attend their sessions. We also have a pre-conference workshop planned on app development using Flutter and have exciting panel and tutorial sessions. We are also honored to have Mr. Kurt Heckman as our keynote and banquet speaker. Mr. Heckman is the President of vCalc, and has served as a Director in the Office of the Secretary in the U.S. Department of Energy from August 2018 to January 2021. He also served in the Office of Science focusing on High Energy Physics and Nuclear Fusion. His talks will be on "Building a Software Company" and "Cutting Edge Advances in Computer Science and Our Futures".

I would personally like to express my sincere thanks to the Conference Committee, Regional Board, Paper Reviewers, Site Chair, and the members of the Math, Stat, and computer science department whose efforts make this conference an excellent forum for our community and students to share their findings and ideas and learn more about our fields. We are always happy to welcome new volunteers, so please let us know if you are interested in joining us. Thank you for being part of the 29th CCSC: MW annual conference.

To all the participants and guests, on behalf of the conference committee and the UW-Stout campus, we hope you enjoy the conference and find it a good opportunity for professional development and for connecting with your colleagues from different schools and industries.

Saleh Alnaeli
University of Wisconsin-Stout
Conference Chair

2022 Steering Committee - CCSC Midwestern Region

Imad Al Saeed, Registrar (2022) ... Saint Xavier University, Orland Park, IL
Saleh M. Alnaeli, Editor (2024) .. University of Wisconsin-Stout, Menomonie, WI
Stefan Brandle, Webmaster (2023) ..... Taylor University, Upland, IN
Mary Jo Geise, Treasurer (2023) ..... University of Findlay, Findlay, OH
Sean Joyce, At-Large (2022) ..... Heidelberg University, Tiffin, OH
Kris Roberts, At-Large (2022) Ivy Tech Community College, Fort Wayne, IN
David L. Largent, Regional Representative (2023) ..... Ball State University, Muncie, IN
Saleh Alnaeli, Conference Chair . University of Wisconsin-Stout, Menomonie, WI
Grace Mirsky, Past Conference Chair ..... Benedictine University, Lisle, IL

2022 CCSC Midwestern Conference Committee

Saleh M. Alnaeli, Conference Chair ..... University of Wisconsin-Stout, Menomonie, WI
Lucy La Hurreau, Vice-Chair ..... Ivy Tech Community College, IL
Diane Christie, Site Chair ... University of Wisconsin-Stout, Menomonie, WI
Saleh Alnaeli, Authors ..... University of Wisconsin-Stout, Menomonie, WI
Cyrus Grant, Nifty Tools and Assignments ..... Waukesha County Technical College, Pewaukee, WI
Cathy Bareiss, Panels, Tutorials, Workshops .. Bethel University, Mishawaka, IN
Imad Al Saeed, Papers ..... Saint Xavier University, Orland Park, IL
Grace Mirsky, Past Conference Chair ..... Benedictine University, Lisle, IL
Paul Talaga, Programming Contest Co-Chair ..... University of Indianapolis, Indianapolis, IN
Md Haque, Programming Contest Co-Chair ..... University of Indianapolis, Indianapolis, IN
David Largent, Publicity ..... Ball State University, Muncie, IN
Imad Al Saeed, Registrar (2022) ... Saint Xavier University, Orland Park, IL
Deborah Hwang, Co-Registrar ..... University of Evansville, Evansville, IN
Stephen Brandle, Speakers Chair ..... Taylor University, Upland, IN
Scott Anderson, Speaker Co-Chair ..... University of Southern Indiana, Evansville, IN
Mary Jo Geise, Treasurer (2023) ..... University of Findlay, Findlay, OH
Paul Gestwicki, Student Showcase ..... Ball State University, Muncie, IN
Matt Green, Student Showcase Co-Chair ..... Waukesha County Technical College, Pewaukee, WI
Kris Roberts, Two-year College Liaison Co-Chair ..... Ivy Tech Community College, Fort Wayne, IN
Takako Soma, Vendors ..... Illinois College, Jacksonville, IL
Stefan Brandle, Webmaster ..... Taylor University, Upland, IN
Jeff Lehman, Work-in-progress Chair .Huntington University, Huntington, IN

Cutting Edge Advances in Computer Science and Our Futures

Banquet Address

Kurt Heckman

Eastern Nazarene College
Quincy, MA 02170
kurt.heckman@vcalc.com

Mr. Heckman will provide a lecture and Q&A session on cutting edge advances in computer science and their cumulative potential for good and abuse in society. The core discuss will circle around the topics of Artificial Intelligence, Super Computers, Quantum Computing, RFI chips, Digital Currencies, Global WiFi and Brain-computer Interfacing.

The goal is to look at the conjunction of these rapidly advancing technologies and challenge the computer engineering students and professionals to envision the amazing opportunities these technologies can provide while drawing attention to the possible hazards to society.

Building a Software Company

Keynote Speech

Kurt Heckman

Eastern Nazarene College
Quincy, MA 02170
kurt.heckman@vcalc.com

Mr. Heckman will provide a lecture and Q&A session on the keys to success and failure that he has experienced in his career as businessman, scientist and entrepreneur of several software engineering companies (Sycamore, Buttonwood, Buttonwood iNet and vCalc).

The lecture will discuss how to position yourself to launch, build and sell software companies. The lecture will also discuss the differences between a software services company and a software product company with the benefits and challenges of both.

Examine the Use of Virtual Worlds for Modeling, Prototyping, and Testing Purposes in a Classroom Setting*

Imad Al Saeed
Computer Science Department
Saint Xavier University
Orland Park, IL 60462
alsaeed@sxu.edu

Abstract

The purpose of this study is to examine the effectiveness of using an open-source virtual worlds environment to enhance programming, modeling, simulation, and testing learning of mobile development applications within the United States Universities. This study included a target population of 15 graduate students each who enrolled in Software Engineering courses at Saint Xavier University. Kotlin mobile app programming language has been used as the main programing language to develop mobile application using Android studio and a combination of Java and C++ programing languages used for simulation and prototyping in Virtual world. The general findings indicated that the using an open-source environment enhanced students learning of new modeling and testing techniques and improve their grades.

1 Introduction

Testing the air brake system for freight cars is an essential function in the railroad industry. No freight car would be attached to the train without passing that test by using special air brakes test device called Automatic Single Car

Test Device (ASCTD). ASCTD is a computerized testing device designed to perform freight car air brake tests in accordance with the Association of American Railroads (AAR) Standard test code S-486. A few years ago, the AAR's changed the standard test code for the S-486 specification to a new standard test code called the S-4027 specification [7]. Currently, all manufacturers' companies such as New York Air Brake, WABCO, and Graham White are either updating their testing devices or designing a new one to support the new S-4027 test code. Normally, railroad companies buy this testing equipment and set up classes to train their employees on how to use it. They use real cars and real testing devices to perform this training at their facilities. Railroad equipment is too expensive; for example, the cheapest ASCTD device is a round $10,000.

All railroad companies set up new classes to train their employees again to perform the freight car test according to the new test code standard S-4027 [7]. Virtual worlds are a unique environment that would assist in eliminating the entire cost associated with planning, modeling, and training the air brakes test in real-life [3]. As a result, a strategic purpose of using virtual worlds is for modeling and training the individuals and offering a general framework based on existed and proved methodologies to be adopted for evaluating their efficacy [5] and introducing unique problems/scenarios that could be very hard and expensive to do in real life.

Develop an alternative low-cost specific 3D virtual training environment can help in enhancing practice skills quickly, safely, low-cost, and in a way that makes the learning points obvious. Students quickly adopted an innovative idea to develop a mobile app to be used to control ASCTD using Kotlin programing [1] using Android studio. They prototyped a training simulation for freight car air brake test using second life (SL) and allowed other students to test their simulated training environment. SL is one of the widely popular web 2.0 tools [4] that can be employed to reach that goal.

2 Why Virtual Worlds?

Currently, virtual worlds are widely used for training and education purposes to facilitate trainees' learning activities [3]. Virtual worlds allow users to create their own avatars, which are referred to their residents and offer them the platform they need to interact with each other easily. Virtual world is a playground for imagination and expands the boundaries of users' creativity in exploring, defining, creating, designing, modeling real environments, building, coding, sharing and recording facilities, performing, and collaboration [6]. Virtual worlds can be very effective and cost-efficient environments that can provide a new methodological framework that supports training purposes [5]. Virtual worlds are a playground to simulate real world applications for training

purposes, but there is a chance it may not be able to simulate overall scenarios with small details that can be involved in real systems because of the limitation boundaries of virtual worlds. This paper lays out the strategic use of virtual worlds within the railroad industry for air brake test training purpose and offers a framework that could be employed to facilitate modeling and testing process.

3 Strategic Use of Virtual Worlds for Modeling and Training Purposes

The evolution of the current technologies and software engineering has allowed for new ways of modeling and exploring many different applications. "Virtual worlds are synthetic representations of reality that are focused on the experience that the users of these worlds have. Virtual worlds take place in real-time and can be used by distributed groups of large numbers of users, and are immersive, and interactive" [2]. The air brake test concept shows prospects for strategic use of virtual worlds because the virtual world applications allow collaborative use of three-dimensional spaces, which are used for modeling and learning purposes in different domains. Chen et al, argues that "the main strengths of virtual worlds could be generalized in the areas of communication, visual expression of information, collaboration mechanisms, interactivity, and entertainment" [2]. As a result, virtual worlds have the potential of offering new capabilities for users to enhance and promote modeling and learning in a number of testing scenarios for air brake tests. Virtual worlds can be very effective for modeling, learning, and training, and are focusing upon the strengths of virtual worlds for supporting training distributed groups in their use of air brakes test. The strategic purpose of using virtual worlds is for modeling and training the individuals and provides a framework based on existing methodologies to be employed for evaluating their efficacy. SL will allow these countries to train in a virtual environment including allowing these countries to be trained by others without a need to travel.

4 Methodology

Hypothesis For this study, the general hypotheses were:

> **H1**: Virtual words enhanced students modeling and testing skills.
> **H2**: There should be a class teaching students more about virtual worlds taught as one of the main courses within all of the computer science department at United States Universities.

H3: The course focused heavily on helping students understand the general principles of simulation, prototyping, and modeling using virtual words.

5 Variables

Learning variables were related to whether students majored in subspecialties of computer science and/or computer engineering technology, the number of programing courses students had already taken, the students' experience in Kotlin and/or Java programing languages, and the students' experience with open course virtual environment such as Second Life. In addition, time on task could be an important learning variable.

6 Logistics of Data Analysis

According to Sharp, the first step in data analysis is identifying the recurring patterns or themes [8]. The researcher collected the data from the questionnaires, students' feedbacks, experimental study, and post- surveys, and used descriptive statistics analysis by importing the data into the (SPSS) where percentages and means were generated and included in the analysis to supplement and clarify quantitative analysis. SPSS is a powerful tool, because it can sort through complex relationships between data and is thus a very effective and efficient way to analyze data.

7 Experiment

The purpose of the modeling activities is to offer a framework that could be used for supporting a larger collaborating modeling build process to explore, imagine, and create an innovative approach to simulate a 3D environment such as an air brakes test environment for training purpose. This modeling framework could be used to construct an action plan for implementing the concept design. Figure 1 shows the general steps that could be followed and implemented for performing any design build. The corresponding steps are:

1. Initial innovative active prospect.
2. Enhance the innovative prospect.
3. Knowing the design requirements, possible tradeoff, and setup an initial prototyping plan.
4. Refine the early concept in an early prototype.
5. Collecting feedback on early design model.
6. Enhance the early design model.

7. Complete the final design.
8. Implement the usability and interaction.

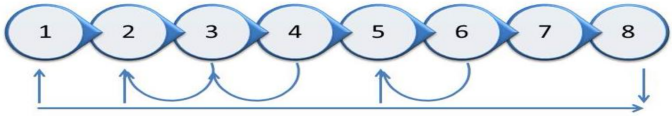


Fig 1. General modeling process.

The three segments to the design framework that described the proposed design activities are:

> **Segment 1**: Explore -Spending more time visiting various locations in virtual worlds such as Second Life (SL), OpenSim, etc. Exploring the initial idea behind promising concepts and come across some tips that could help in modeling application systems.
> **Segment 2**: Imagine - Refine the early concept and present the project concepts in an early prototype. Engage other people for collecting feedbacks on early design model.
> **Segment 3**: Create - Create an action plan for implementing the design model using SL and perform a usability and interaction with the people, industry, or organization, which is interesting in a design model.

By following the proposed design activities framework, a virtual training environment for air brake test was modeled and prototyped in SL using a combination of Java and C++ programing languages. The hardware part was prototyped by building a 3D virtual yard environment for the air brakes test (see Fig 2(a)), and one ASCTD attached to a car through the brake pipe house from one side and to the air supply from the other side (see Fig2 (b)), while the software part was prototyped by building the interface software that contains the main initial setup page of the mobile app and the main interface testing screen. The main simulation logo contains the note card that includes some information of what to do to perform each task within the air brake test.

Ten participants were engaged from a Software Engineering class for collecting feedbacks on the early design model. Below are top four feedbacks from four participants:

> **Participant 1**: Add a signal communication indicator to make indication that there is a communication between the main computer and the ASCTD.

Participant 2: Add a registration and visitor counter to count the number of people that already visited the air brake test website and ran the test.
Participant 3: Make a PowerPoint presentation and keep in the design location area that includes all the testing sequences steps. In this way any user can read these instructions and easily perform the test.
Participant 4: Add security log-in issues such as username and password to the interface software.

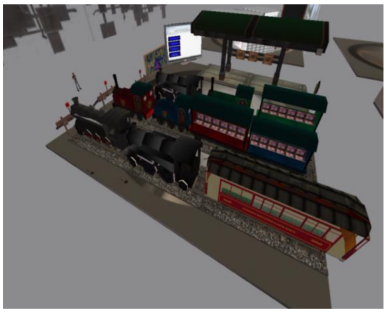


Fig2 (a) Simulation of a virtual train yard in SL

Fig2 (b) Simulation design of the air brake test in SL

The training simulation was improved according to the participant feedback.

8 Results

The demographics information for students in the post-Survey were: (84%) have great knowledge with Java programming and they easily transition to Kotlin programing, (31%) was reported that they learned Kotlin from scratch, and (100%) have never used any open-source virtual environment before. The researcher that collected evidence helped him approve H1, H2, and H2 by conducting an online post survey using survey monkey and class observation and presentation. The results showed there were 90% of the participants from the post-survey indicated that virtual words enhanced students modeling and testing skills. Also, 72% of the participants indicate that there should be a class teaching students more about virtual worlds as one of the main courses or as an

elective course within all off the computer science department at United States Universities, while a very limited number of the participants were neutral in their responses, and no one disagreed with that claim at all. A high percentage of the participants (87%) from the online post-survey indicated that the course focused heavily on helping students understand the general principles of simulation, prototyping and modeling using virtual worlds. In addition, those participants (95%) from the online post-survey were very interested in building their team further with researchers from other University departments who can benefit from the computational speedups and larger capacity afforded for testing computerized based systems. The results reflected from the students' post-survey showed 82% of the students recommend this course to their peers that are interested in simulation and prototyping, which added more validity to the results. Finally, a classroom observation and students' presentations finding also strongly supported and added more validity to the results reflected from the online post-survey.

9 Conclusion

The online air brakes test system concept shows prospects for a strategic use of virtual worlds because the "virtual world applications allow collaborative use of three-dimensional spaces which are used for modeling and learning purposes in different domains" [2]. The main strengths of virtual worlds could be realized in the areas of communication and collaboration mechanisms, and visual expression of information, interactivity, and entertainment. In conclusion, virtual world simulation could provide users with a higher level of realism of the air brakes test simulation, enhance practice skills quickly and safely, and cut the costs of the real training, and in a way that makes the learning points obvious. In addition, the general framework offered allows for greater flexibility for different training scenarios, such as air brake testing scenario. This study presents evidence that a strategic use of virtual worlds within the railroad industry for training purposes and provides the segments to the design framework that described the proposed design activities.

10 Future Study

The future study shall be focused on the usability and interaction testing technique to be used to evaluate air brake test simulation by testing it with users to further examine the strategic concept and assess its impact. SL can be used for performing a usability test for air brake testing, because it gives direct input on how real users will use the system in the real world. Usability testing focuses on measuring the engineering design of the air brake test system interface capacity to meet air brake testing rules according to AAR. Setting

up a usability test in 3D virtual worlds such as SL involves carefully creating a testing scenario(s) for simulating the realistic situation, wherein the person should perform a list of tasks using the air brakes test interface system being tested while observers use think aloud protocol, eye tracking, and take notes to gather feedback.

References

[1] A modern programming language that makes developers happier. Retrieved on May 05, 2022 `www.Kotlinlang.org`.

[2] Yiyu Cai, Wouter Van Joolingen, Zachary Walker, et al. *VR, Simulations and serious games for education*. Springer, 2019.

[3] Yung-Fang Chen, Genaro Rebolledo-Mendez, Fotis Liarokapis, Sara de Freitas, and Eleanor Parker. The use of virtual world platforms for supporting an emergency response training exercise. 2008.

[4] Sara De Freitas. Serious virtual worlds: A scoping study. 2008.

[5] Horácio Gaspar, Leonel Morgado, Henrique Mamede, Teresa Oliveira, Baltasar Manjón, and Christian Gütl. Research priorities in immersive learning technology: the perspectives of the iLRN community. *Virtual Reality*, 24(2):319–341, 2020.

[6] Leonel Morgado, Hugo Paredes, Benjamim Fonseca, Paulo Martins, Álvaro Almeida, Andreas Vilela, Bruno Pires, Márcio Cardoso, Filipe Peixinho, and Arnaldo Santos. Integration scenarios of virtual worlds in learning management systems using the MULTIS approach. *Personal and Ubiquitous Computing*, 21(6):965–975, 2017.

[7] Association of American Railroads. Retrieved on may 05, 2022. `http://www.aar.org`.

[8] H Sharpe, Y Rogers, and J Preece. *Interaction design: beyond human-computer interaction 2nd ed.* John Wiley Sons, Inc.

Lessons Learned Teaching Programming During the Pandemic*

Jean Mehta
Saint Xavier University
Chicago, IL

Abstract

The pandemic prompted universities in the US to move to remote learning, causing many of us who teach programming in a face-to-face pedagogy to question whether we could be successful in a remote format. While not a research paper per se, this is an experience report describing the transition to online teaching due to the pandemic. The paper discusses pedagogical adaptations that needed to be made in order to maintain excellent retention rates during the pandemic, and which of these adaptations should be retained with a return to the face- to-face environment.

1 Background

Saint Xavier University, located in Chicago's southwest side, was founded by the Sisters of Mercy in 1846 to provide high-quality education to underserved populations while preparing them to serve their communities with wisdom and compassion. This mission is expressed in the contemporary era through intentional, academically-rigorous degree programs designed to prepare scholars with the competence and creativity necessary to meet labor market demands and bring benefit to the wider community. The student population is majority-minority – 42% of students identify as Hispanic/Latinx, 11% as African American and 9% as other underrepresented minorities. The computer science department currently has 100 – 120 undergraduate students in

two majors, Computer Science (CS) and Computer Information Systems (CIS), of which 67.5% receive state and federal aid. Many are commuters, most work; some have more than one job.

2 Pre-Pandemic

The department carefully monitors retention in CS1 and CS2, both of which were traditionally taught on campus in two-hour blocks – two hours each day, two days a week. We use the flipped classroom methodology; the use of an electronic textbook [5] enables us to assign exercises that must be done prior to class. This allows for a small amount of instruction and/or clarification in class, but the majority of time is given to students completing small programming projects while a tutor and myself circulate giving help. In a prior study the retention rate had successfully been increased from about 70% (the national norm) to close to 90% using the above pedagogy.

Then came the pandemic and all teaching/learning moved to remote. I had always claimed that I could not successfully teach programming online, and would not be able to maintain these retention rates. Moreover, I was not alone in this belief. Pre pandemic data showed an alarming drop and failure rate. In a review of literature on retention in online courses, Bawa [1] concludes that "Online courses have a 10% to 20% higher failed retention rate than traditional classroom environments." Glazier [3] puts the range at 5% to 35%.

I wanted to continue with my successful teaching paradigm, but worried about my ability to do so in a remote environment. Given that we moved to the online platform mid-semester (Spring 2020), and that the students were then given the opportunity to take the course Pass/Fail, the results from this first semester were not tracked. The following three semesters were taught in a virtual synchronous manner. This is what I found worked (or not) for myself and my students, and what I took back when returning to the face-to-face environment in Spring 2022.

3 Pandemic semesters (virtual synchronous mode)

Two tutors were hired for each class period, and additional tutors for each day of the week (including Saturday and Sunday) so that students would have a generous amount of support outside the classroom.

A huge number of videos were made (2 or 3 for each class period in each course) presenting examples of programs with my verbal explanation of what I was doing and why. Making these in advance enabled me to edit them and have them ready for students before class for the flipped classroom, and I also recorded my class lecture when appropriate.

Small programming projects (labs) were integrated into the textbook along with their associated unit tests, enabling students to create and execute their programs right in the textbook's programming environment. The unit tests would run their program with different input and the results were then displayed on the student's screen giving instant feedback. Solutions, made available immediately after the assignment's due date, enabled students to see where they made a mistake, and how I approached the problem. During these three semesters we had students who were quite sick with Covid, and we needed to alter the due dates for these students. We learned how to make the solutions available to students depending on their due date, so that we were not holding up the solution for the majority of the class because one student was currently sick.

During the class period I would introduce a new topic, often using a whiteboard. I found it most effective to have a whiteboard on my screen side by side with the programming environment so I could move back and forth between the two. In this way I could mimic the way I taught pre pandemic, using the electronic whiteboard in place of the physical one, and putting my code on my screen in place of the physical screen. However, I found that being able to record and save these was a benefit to students – they could replay the videos when necessary or pause them while they tried something on their own.

So far we had been able to convert our previously successful strategies to the online format, but how to mimic sitting next to students and helping them with their code? Students could post a message to chat (I used Canvas Conferences for my class periods) or private message me or a tutor during class if they wanted to ask a question but didn't want to ask it in the public space. I also made a lot of use of breakout rooms. Then, if the question was more complex or if the student's code needed to be debugged, I (or a tutor) could go into a breakout room with a student and the student could display their screen. This kept our conversations private and the student able to share their screen only with me and not be potentially embarrassed by having to display his screen to the entire class. Students benefit greatly by working with an instructor or tutor at a time they are having difficulty with the course content, and are perfectly poised to learn. The privacy of the breakout room enabled the student to more fully acknowledge inadequate comprehension of the material thereby leading to a better outcome. Moreover, it enabled me to get to know each student better by providing the personal interaction that we lost moving to a remote environment. I also occasionally assigned a program to be completed as a group and put the students in groups in breakout rooms – really so they could have more contact with each other and not be isolated. We also gave them a few minutes to chat as we were aware that they needed to socialize and were feeling very isolated especially at the beginning of the pandemic when there

were no vaccines and people were at home more. Tutors (and I) would move between the rooms in order to elicit conversation and to keep them working as a group.

A survey of CS faculty distributed in 2020 through Computing Research Association (CRA) and SIGCSE listservs received 450 responses [2] in which 74.6% reported they found it hard to implement their preferred teaching style, and 34.9% reported discontinuing active learning. We were pleased to be able to continue with our pre-pandemic teaching style, including active learning. However, the question remained – was it sufficient; would we still be able to keep our pre-pandemic retention rates.

4 Results – what worked

Weekend tutors were a big hit. We had not had tutors over the weekend previously, nor had we considered the possibility, and since all our tutoring had been on campus it had been difficult to find tutors (we are a commuter campus). But now that all our tutoring was online, we had a large pool of students from which to find excellent tutors for both in the (remote) classroom and also outside those hours.

Prior to the pandemic I held office hours on campus and in my physical office for one hour on each of three days. However, usually, nobody came. Remotely, with the help of Zoom, it was very easy to invite all students from all of my classes, and also all my (60+) advisees to each of my extended office hours (two hours, three times each week). Many students visited me, some for help, others just to check in and chat. Several times a student commented that they saw the invitation to a zoom meeting, realized I was available, and decided to connect. Many times we heard that students felt isolated during the pandemic, and I was happy to try to maintain some contact with them outside the classroom as well as during class. In retrospect I realize that this was of benefit to me too as I was also isolated and so increasing my office hours wasn't a burden but rather an opportunity to help/mentor/ or just connect with students.

The biggest surprise for me was finding that breakout rooms were a great advantage over being in the classroom, giving students increased privacy, which they valued. Prior to this semester I would sit beside a student and comment on the code, but everyone around the student could hear. Now, students often privately contacted me or a tutor asking for a breakout room. Once there, the student's privacy was secure. If I saw something that I wanted to tell the entire class I could post it in chat and none of the other students knew who was responsible for this issue because they didn't know who had been in a breakout room with me. Also, on the issue of increased privacy, students

who may have been reluctant to raise their hand to ask a question in class could private message a tutor or me and we could answer privately. If it was something that we felt needed to be addressed to the entire class we could say "someone just asked...." But, of course, without naming the student. Here is a comment from a student regarding breakout rooms "I have anxiety and often times I don't feel comfortable in class when it comes to asking questions or participating. Breakout rooms made me feel comfortable and therefore created an optimal environment for learning."

Many students commented on the videos, saying they were useful and that they could pause and replay them when necessary. Prior to the pandemic, when we were in the classroom, students would occasionally take a photo of the whiteboard if they thought something was interesting, or to be remembered. An isolated photo, they may later wonder why they took it and what it meant. With videos I may still be writing on the whiteboard, but they have it in context.

Other CS educators have found value in videos. From an ITiCSE Working Group Report [4] from 2022 "most faculty agreed recorded lectures were better than in person for at least some of their students: 63.6% selected that their students' "ability to watch recorded lectures at a different time than class time" was better, while 63.1% agreed that students being able to re- watch lecture portions to better understand the material was helpful."

5 Results – what didn't work

Students hid. Originally, I had visions of being able to see all my students on my monitor, but they ALL turned off their cameras. I had not previously taught the students in CS1 and, without seeing their faces, felt that I never got to know them. I also didn't know if they were really there, or had just logged in and then gone away, leaving their browser open. In each class there were two or three students who just didn't respond. I could keep poking them in chat – "How are you doing?", "What are you doing?", "Can I help". But sometimes received no response whatsoever. Invitations to a breakout room sometimes went unanswered.

It was also difficult for me to receive feedback. Without seeing their faces, I couldn't tell if I was speaking too quickly, proceeding too quickly with a lecture, or if I needed to explain more fully. They could private message a tutor or me asking for clarification, or speak up during the lecture, but nothing quite takes the place of being able to look around your class at the students' expressions.

Videos are not always used the way you intend. I spent hours making them, and initially it took me a really long time to edit even a short one. Having made one, and added it to the week's module so that students could watch it

ahead of the lecture, I often found that they didn't watch it at all. In order to encourage them to watch it ahead of class I embedded a short (two or three question) quiz in each video, due prior to class. But then I had to make a copy, minus the quiz, so that students could watch it later for review and not be forced to take the quiz each time. Another issue I found with the videos is that I hadn't minded all the extra time it took to make them because I had envisaged reusing them, but then found that I had recorded my entire screen, including the date and time, and then in subsequent semesters had to (with the help of Camtasia) cut the date and time located at the bottom right of the video.

6 Results

Of course, with such small class sizes any results cannot be statistically significant, but the anecdotal data are suggestive.

CS1. Taught only in Fall semesters. Fall 2018 Fall 2019 were face-to-face, pre-pandemic. A passing grade is A, B, or C.

Semester	#students pass /total#students	percentage
Fall 2018	31/32	96.9%
Fall 2019	30/34	88.23%
Fall 2020 ** remote	46/48	95.8%
Fall 2021** remote	30/34	88.23%

CS2. Taught only in Spring semesters. Spring 2018 Spring 2019 were face-to-face, pre- pandemic. Spring 2021 was remote, Spring 2022 was back on campus and face-to-face. A passing grade is A, B, or C.

Semester	#students pass /total#students	percentage
Spring 2018	29/31	93.54%
Spring 2019	14/15	93.33%
Spring 2020** move to remote mid semester	Retention rates not considered because of the transition	
Spring 2021** remote	33/36	95.8%
Spring 2022 Back on campus	25/25	100%

Pre-pandemic results from online programming courses showed very high drop rates causing us to question whether we could deliver programming courses online. We not only did, but we excelled, managing to reproduce results similar to those in the face-to-face environment pre pandemic. Also note that in Spring 2022, with a return to campus and the implementation of some strategies learned from the remote courses, there was 100% retention in CS2.

7 Moving forward

Our university made the decision to move all courses back to campus beginning Spring 2022. The question now is what did I learn from moving to the virtual environment, and what can be brought back to the classroom.

The need to be more flexible. In conversations with students, I learned more about their individual situations than ever before. They are mostly commuters, some either going directly from campus to the workplace, or vice versa. Some even work a night shift and then come to class. If I need them to watch a video prior to coming to class, I have to make it available to them a few days in advance, preferably a week.

Give immediate feedback whenever possible. I shall continue to be strict on students' meeting deadlines on assignments, but have learned how to make solutions available immediately after the due date for those students who submitted. I also grade immediately, which, combined with the solution, gives them the instant feedback that they need. I have learned that there is no need to hold up a solution for the entire class just because one student is sick.

Increase tutor availability. I will continue to have tutors in the classroom with me, and will provide evening and weekend hours. It will be impossible to have a tutor on hand at the very time a student needs one, but we should be able to cover a few hours each week night and weekend.

Use videos much more frequently. Videos are a great benefit. It was a huge learning curve to get this right, but a video with embedded quiz ensures most of the students watch it prior to class and this helps with a flipped classroom. Prior to the pandemic, I would write a program, add it to the module in our LMS, and then demonstrate it in class. In the remote situation, I did the same, but I also recorded the demonstration. Going forward, I will produce the program and video prior to class and put them into the module. Then in class I will project the code and discuss it. The video will be useful for students who either missed the demonstration, or need to review. I am certainly aware that this may encourage students to skip class, but I am hopeful that it will help provide the flexibility they need. I am in the process of making even more videos (without quizzes) to put on YouTube. Once complete, this will be the entire course, and will supplement the text that we use. Pre pandemic I supplemented the text with my own notes, but these were very wordy and

without visuals. I will replace these with videos, which can be a resource at any time, but in particular for review before a test.

8 Conclusion

During the pandemic we were able to maintain our pre-pandemic retention rate. In the future, back on campus but having learned from that experience, I am hopeful that we can improve upon it.

References

[1] Papia Bawa. Retention in online courses: Exploring issues and solutions — a literature review. *Sage Open*, 6(1):2158244015621777, 2016.

[2] Betsy Bizot, Ran Libeskind-Hadas, Susanne Hambrusch, Jim Kurose, Lori Pollock, Nancy Amato, CRA CERP Team, et al. Results of a summer 2020 survey of computer science faculty: The transition to online teaching last spring and planning for the fall. *Computing Research Association*, 2020.

[3] Rebecca Glazier. A shift to online courses this fall could lead to a retention crisis. `https://www.edsurge.com/news/2020-07-06-a-shift-to-online-classes-this-fall-could-lead-to-a-retention-crisis`.

[4] Angela A Siegel, Mark Zarb, Bedour Alshaigy, Jeremiah Blanchard, Tom Crick, Richard Glassey, John R Hott, Celine Latulipe, Charles Riedesel, Mali Senapathi, et al. Teaching through a global pandemic: Educational landscapes before, during and after COVID-19. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*, pages 1–25. 2021.

[5] Zybooks. Programming in Java with Zylabs. `http://zybooks.zyante.com`.

A Cost Estimation Model for Scrum Projects*

Xingzhan Feng and Kasi Periyasamy
Department of Computer Science
University of Wisconsin-La Crosse
La Crosse, WI 54601
xingzhan0312@gmail.com kperiyasamy@uwlax.edu

Abstract

One of the daunting tasks of software developers is to estimate the development cost of a new software product. Most cost estimation models use the set of requirements for the product as the starting point. Function-point, COCOMO and use case-based cost estimation models belong to this category. These models assume that the requirements of the product are fairly rigid. However, with the advent of agile-based software development methods, the requirements keep changing during the development process. Therefore, traditional cost estimation models need to be refined to accommodate changes in requirements. The refinements should reflect the changes in cost when the requirements change. In this paper, we describe a cost estimation model for projects that use scrum, a popular agile method. The model uses the requirements of the new software product, written in the form of user stories, as the primary source. The cost is adjusted every time the requirements are changed or new requirements are introduced. We have also developed a project tracking tool for scrum projects in which this model has been implemented. The model was applied to academic projects developed by graduate students; the results indicate that estimations are fairly reasonable.

1 Introduction

Cost estimation of the development of a new software product is a crucial and daunting task. A lot of cost estimation models were reported in the literature,

but many of these models became obsolete because of changes in technology and development methods. Most models, including those that are based on use cases, predominantly use requirements specifications as the source for initial cost estimation. Consequently, they all rely on well-structured documents to specify the requirements. For example, function-point model [4] requires the collection of data inputs, data outputs, number of inquiries and the structure of internal storage space. These elements can be extracted from the requirements specification. Similarly, the use case-based approach used in [9] requires documentation associated with the use case diagram that describes the actors, input and output parameters and exception scenarios of each use case. Since agile methods use mostly unstructured requirements specification (e.g., user stories used in scrum projects), it is difficult to use the cost estimation models published in the literature for agile methods. Besides, as evident from the agile manifesto [2], a welcoming aspect of any agile method is frequent changes in requirements. Because of these changing requirements, any initial cost estimation arrived at using the cost estimation models based on traditional methods would not be valid. Therefore, the cost estimation model for agile methods should be flexible enough to dynamically estimate the cost whenever requirements change.

1.1 Related Work

Most cost estimation models published in the literature fit well for traditional software development using waterfall and incremental prototyping models. Research on cost estimation models for agile software development is still in its infancy. Rosa and others [11] discuss cost estimation models for agile processes in initial phases of development. They focus more on getting an initial estimation based on the problem at hand. This approach is somewhat similar to traditional models for cost estimation such as COCOMO model or Functional Point model. Kang and others [5] used a model-based approach in which they develop an initial model which is dynamically updated as the development progresses. This approach seems to work better than Rosa and his team, but the work involves additional tasks to update the model and the cost estimation process. Popli and Chauhan [10] used a sprint-point based technique which is like use-case based approach used by other researchers [9]. Their approach is based on the sprint tasks and helps estimate the cost for each sprint separately. Some researchers [6, 7] believed that the cost estimation of an agile process continues to swing due to technical debt that occurs during each sprint. Therefore, they focused on their cost estimation algorithms based on calculating technical debt and thereby estimating the overall cost. Adnan and Afzal [1] used a scrum ontology model and description logic with multi-agent to gather knowledge from various activities that occur in a scrum-based project and use

this information for guiding the scrum master. Gandomani and others [3] discuss why the 'Planning Poker' approach (one of the commonly used techniques for effort estimation in agile methods) is not reliable. They assert that more concrete information is needed than consensus or averaging the size of user stories used as a source for initial cost estimation.

Our work reported in this paper uses a technique like the one used by Popli and Chauhan [10], and hence is meant for scrum projects. We enhanced Popli and Chauhan's model with a repository-based system. To start with, an initial cost is derived from the user stories. As the user stories evolve during the development of the product, so is the cost associated with them. This process help visualizing the evolution of the cost as user stories are changed and new ones are added. At any point in time, the development team may reject addition of new user stories if the cost exceeds far from expectation.

2 The Model

In a scrum project, the requirements of a software product are written in the form of user stories. To illustrate, the user story to login into a system will be given as "*As a user, I would like to login into the system so that I can gain access to the features or services provided by the system*". Each user story is associated with a set of sprint tasks that describe the implementation details of the user story. For example, the set of sprint tasks associated with the above user story can be given as below:

- Develop the GUI for the user to input login credentials such as username and password.
- Develop backend code to validate the inputs.
- Develop backend code to access the named account from database and verify the credentials.
- Send confirmation message to the user for successful login or error message to the user for any failure.

Knowing the set of sprint tasks to be implemented, we can now estimate the cost to implement each sprint task, and add them up to get the cost to implement the user story. If we know the set of user stories to be implemented, the cost for the project will then be the sum of costs for all user stories.

Formally, let u_1, u_2, ..., u_n be the user stories for a scrum project. Let s_{i1}, s_{i2}, ..., s_{ik} be the sprint tasks associated with the i^{th} user story. Notice that the number of sprint tasks (in this case k) will vary for each user story. Let $costU(u)$ be a function that estimates the cost of the user story u, and

$costS(s)$ be a function that estimates the cost for the sprint task s.
$costU(u) = \sum_{i=1}^{k} costS(s_{ui}) + \epsilon_u$
where ϵ_u defines the cost to integrate and test all the sprint tasks, to realize the user story u. Here, we assume that the cost of unit testing a sprint task is included in the cost function $costS$. Finally, let $costP(project)$ be the cost of the project which can be defined as
$costP(project) = \sum_{j=1}^{n} costU(u_j) + \epsilon_{project}$
where $\epsilon_{project}$ corresponds to the initial cost for setting up the development environment such as a server, database, utilities etc. This is important because each project may use different set of utilities, tools, technologies and software libraries. Even for well experienced software developers, setting up the development environment takes considerable time.

Since the cost of a project is defined in terms of the cost of user stories to be implemented, the cost of the project can be split into two parts - *stable cost* and *unstable cost*. The *stable cost* refers to the cost obtained from those user stories that are currently implemented and approved by the project's stakeholders. In contrast, the *unstable cost* refers to the cost obtained from the user stories that are not yet implemented. To start with, *stable cost* will become zero and will gradually increase during each sprint. When the project is finished, *stable cost* gives an indication as to whether the product has been implemented within the expected time. Moreover, these two costs should be monitored throughout the project duration. Whenever user stories are changed or newly introduced (happens all the time in scrum), the two costs should be re-evaluated.

2.1 Estimating the Cost of a Sprint Task

As mentioned in the previous section, the cost of a project depends on the cost of user stories which, in turn, depends on the cost of implementing the corresponding spring tasks for each user story. We propose two methods to compute $costU(u)$ of the user story u based on its sprint tasks.

2.1.1 Heuristic Method

Let u be a user story and s_1, s_2, ..., s_k be the sprint tasks associated with u. Based on the skills of the development team members and velocity of the team (referring to the average number of user stories or sprint tasks implemented in each sprint), $costS(s)$ of a sprint task s may take from t_{min} to t_{max} hours. For example, in students' academic projects, t_{min} may be 4 hours and t_{max} may be 12 hours, whereas in an industrial setting, t_{min} may be 1 hour and t_{max} may be 6 hours. Having an estimation of t_{min} and t_{max}, $costS(s)$ can be computed as the average of the two limits; i.e., $costS(s) = (t_{min} + t_{max})/2$.

It should be noted that t_{min} and t_{max} vary for each sprint task. Further, the estimation of t_{min} and t_{max} involves the skills of the development team members, the velocity of the team, and the number of user stories selected to be implemented in the current sprint. Some of these factors may keep changing during subsequent sprints. For example, the development team gains more experience with the tools, technologies and the application domain, and hence their skills improve continuously. As the team gains more experience, their velocity may also improve. Therefore, t_{min} and t_{max} for each sprint should be re-evaluated for every sprint task in every sprint.

2.1.2 Pattern Matching Method

In this method, $costU(u)$ of a user story u is calculated by matching a similar user story from a previously completed project. In order to do so, previous projects and their complete details must be stored in a data store. Let $newU$ be a new story and $newS_1$, $newS_2$, ..., $newS_k$ be the set of sprint tasks associated with $newU$. When the user creates $newU$, it is matched against all user stories from previously completed projects that are already stored in the system . Since a user story is expressed as an informal sentence (refer to the example in the previous section), matching of user stories can be done using natural language processing algorithms. Let $oldU_1$, $oldU_2$, ..., $oldU_n$ be the user stories that closely match with $newU$. Since the user stories $oldU_1$, $oldU_2$, ..., $oldU_n$ come from previously completed projects, the time taken to implement each one of these user stories is available and hence the time taken to implement $newU$ can be roughly estimated. This can be done either by computing the averages of completion times for the user stories $oldU_1$, $oldU_2$, ..., $oldU_n$, or by choosing the completion time of the best matching user story among them. In the latter case, the rough estimation can be fine-tuned by matching the sprint tasks of the old user story with those of the new one.

3 Project Tracking Tool

The authors have developed a project tracking tool for scrum projects. The primary function of this tool is to create and maintain the artifacts of scrum projects (user stories, sprint tasks and test cases) and so the tool helps software developers monitor the progress of a scrum project. In addition, the cost estimation model described in the previous section has been implemented in the tool. A previous version of this tool included a machine learning model [8], but the current version includes the model described in this paper. The user of the tool can choose either the heuristic method or the pattern matching method to compute the cost of each new user story. To support the pattern matching method, the tool includes a repository of all artifacts of previously

completed projects. The tool maintains both the stable and unstable costs of each project. An administrative user of this tool can import these artifacts from external projects as well as from those projects that were completed using this tool. While importing from external projects, the tool requires the artifacts to be formatted in a particular way that is acceptable by the tool.

When a new project is created, the initial cost estimation can be obtained by inputting all user stories of the new project. The tool, in turn, compares each new user story with those in the repository and selects the closely matching user stories using the pattern matching method. If, for any new user story, the tool does not find any matching user story from the repository, then the tool will suggest to include the sprint tasks for the new unmatched user story and then use the heuristic method to compute the estimated completion time of that user story. The estimated cost for the new project will then be the sum of estimated costs of all user stories.

3.1 Cost Drivers

The rough estimation of completion time for user stories as described in the previous section is considered to be 'unadjusted' because the completion time of a user story by two different development teams might be different. This may be due to the expertise or skill set of the developers, the tools and technology they use, and their understanding of the application domain. In traditional cost estimation model, this kind of discrepancy is adjusted through cost drivers. These are environmental and technical factors that fine-tune an unadjusted estimation. The authors also developed a questionnaire that represent these cost drivers. Each development team is expected to complete this questionnaire at the beginning of the project. The tool will then evaluate the contribution of cost drivers towards accurate estimation. The questionnaire is given in the appendix. Using the answers provided for the questionnaire, the tool will calculate a factor due to cost drivers and then multiply this factor with the unadjusted cost computed earlier. Due to space limitations, the details of calculation involving cost drivers are not given in this paper.

3.2 Limitations

The scrum project tracking tool is subject to the following limitations:

- In order to use the pattern matching method described in section 2.1.2, the tool requires a repository of previously completed projects. The bigger the size of this repository, the more accurate will be the initial rough estimation. However, this will also create a storage problem as the repository grows bigger.

- If artifacts from external projects are imported into the tool, the artifacts must be in a format fixed by the tool.

4 Conclusion

Cost estimation of a yet to be developed software product is a crucial task. There were several cost estimation models published in the literature, but they seem to vary based on changing technologies, software development life cycle models and the expertise of development team members. Traditional cost estimation models work well for software products whose requirements are stable. This is due to the fact that the models use requirements of the product as the primary source. These models are inadequate for development approaches that use agile methods because the latter welcome changing requirements during product development. In this paper, the authors have proposed a cost estimation model for scrum projects. Scrum is one of the popular agile methods. The proposed method gives an estimation of the product which will be re-evaluated whenever the requirements, in the form of user stories, are changed or new requirements are introduced.

The authors have also developed a tool to maintain the artifacts of a scrum project, and to track the progress of the project. This tool implements the proposed cost estimation model so that the developers can also estimate and monitor the cost of new software projects. In addition, the tool includes a repository of previously completed scrum projects. The tool, at first, computes an unadjusted cost for a new project. A cost driver module is built into the tool which will later fine-tune the unadjusted cost giving a more accurate cost information. The cost drivers are computed from the answers for a set of questions given to the developers. These questions are based on the environmental factors used by the development team and personnel questions related to the skills of the developers. The tool has been tested with academic projects developed by graduate students in a software project management course.

References

[1] Muhammad Adnan and Muhammad Afzal. Ontology-based multi-agent effort estimation system for scrum agile method. *IEEE Access*, 5:25993–26005, November 2017.

[2] Kent Beck *et al.* Agile manifesto.

[3] Taghi Javdani Gandomani, Hamidreza Faraji, and Mahsa Radnejad. Planning poker in cost estimation in agile methods: Averaging vs consensus.

In *5th International Conference on Knowledge-based Engineering and Innovation (KBEI'19)*, pages 66–71, February 2019.

[4] IFPUG. International function point user group.

[5] Sungjoo Kang, Okjoo Choi, and Jongmoon Baik. Model-based dynamic cost estimation and tracking method for agile software development. In *IEEE/ACM 9th International Conference on Computer and Information Science*, pages 743–748, USA, August 2010. IEEE Computer Society.

[6] Antonio Martini and Jan Bosch. The magnificent seven: Towards a systematic estimation of technical debt interest. In *XP'17: Proceedings of XP 2017 Scientific Workshops*, New York, NY, USA, May 2017. Association for Computing Machinery.

[7] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. An empirical model of technical dept and interest. In *Second Workshop on Managing Technical Debt, part of International Conference on Software Engineering (ICSE 2011)*, pages 1–8, May 2011.

[8] Kasi Periyasamy and Josh Chianelli. A project tracking tool for scrum projects with machine learning support for cost estimation. In *29th International Conference on Software Engineering and Data Engineering (SEDE 2020)*, volume 76, pages 86–94. EPiC Series in Computing, October 2020.

[9] Kasi Periyasamy and Aditi Ghode. Cost estimation using extended use case points (eucp). In *International Conference on Computational Intelligence and Software Engineering (CiSE'09)*, pages 2556–2560, December 2009.

[10] Rashmi Popli and Naresh Chauhan. A sprint-point based estimation technique in scrum. In *International Conference on Information Systems and Computer Networks (ISCON'13)*, pages 98–103, March 2013.

[11] William Rosa, Raymond Madachy Bradford Clark, , and Barry Boehm. Early phases cost models for agile software processes in us-dod. In *IEEE/ACM International Symposium on Empirical Software Engineering and Measurement*, pages 30–37, USA, November 2017. IEEE Computer Society.

Integrating The OpenMind Platform into a Human-Computer Interaction Elective*

Paul Gestwicki
Computer Science
Ball State University
Muncie, IN 47306
pvgestwicki@bsu.edu

Abstract

This experience report details the integration of The OpenMind Platform into a design-focused, upper-division undergraduate elective course on Human-Computer Interaction. The OpenMind Platform provides a series of interactive online activities through which users learn about how the mind works and what that means for engaging in constructive dialog. The learning objectives of The OpenMind Platform coincide with many HCI learning objectives. Particular points of synergy include confirmation bias, motivated reasoning, intellectual humility, and the value of diverse perspectives.

The experimental course was online and asynchronous. It was offered during the Fall 2020 semester, which was a season of political, cultural, and global health upheaval. The student response to the integration of The OpenMind Platform was overwhelmingly positive, and many proclaimed this to be one of the most impactful parts of the class. The complete integration plan is provided in an online reference, although the specific structure of The OpenMind Platform has changed since the semester described in this paper.

1 Introduction

One of the outcomes of studying human-computer interaction (HCI) in undergraduate Computer Science is to learn that the user is different from the developer. Yet, most student software development has no real end-users: students are programming in order to learn the craft or to demonstrate understanding of a related concept. Through HCI and software engineering coursework, students learn that understanding users is a prerequisite to being able to solve their real problems.

It goes without saying that 2020 took some unexpected turns. Like many instructors, I had to redesign my Fall courses for online asynchronous delivery—a format I had never considered using and would not have chosen for myself. One such course was an upper-division HCI elective. The usual community-engaged projects and face-to-face activities were made impractical by the global pandemic, and so a fundamental redesign was required. The irony of studying HCI in an environment that required computer mediation of human interaction was not lost on me nor my students.

The unexpected modality provided new opportunities to think about the alignment of student activities and course goals—in particular, the goal of helping students understand the importance of empathy and humility toward effective collaboration with diverse audiences. After some investigation and consideration, I integrated The OpenMind Platform[1] into the course. This platform provided a sequence of five interactive online lessons that were designed to help people communicate constructively across differences. That is, it helps people learn to disagree productively. The OpenMind Platform draws upon established, empirically-validated social scientific research while also incorporating wisdom from traditional philosophical and religious sources. Major topics of the OpenMind Platform lessons include confirmation bias, moral foundations theory, intellectual humility, diversity, and cognitive distortions. The project's website provides a wealth of resources, both popular and academic.

The proximal goal of the integration was to support students' learning of HCI principles. At the same time, I hoped that students would recognize that these ideas—both in HCI and from OpenMind—have applicability well beyond the course. Consider, for example, that teaching students to productively disagree has become increasingly important. According to the Fall 2020 Campus Expression Survey, over 60% of college students found that the campus climate prevented them from saying what they believed, and the percentage of reluctant students increased from 2019 [22].

[1] `https://openmindplatform.org/`

2 Context and Background

The course under consideration is an upper-division elective within a Computer Science department. It is taken by majors as one of several directed electives, and minors sometimes enroll as their final elective. Prerequisite coursework includes three programming-intensive courses as well as discrete mathematics. The department's syllabus states that, by the end of the course, the student should be able to: explain why user-centered product development is important; explain principles for design of user interfaces; create a simple graphical user interface and justify its usability; plan and execute a usability evaluation; and describe model-view separation and explain its value. These build upon the core learning outcomes recommended by CS2013 [1] and those presented in drafts of the upcoming revision. The syllabus empowers different faculty who teach the course to have different emphases. My approach is to teach the HCI course with an emphasis on design. In particular, I teach that design is a goal-directed process, and that a design's quality is determined by fitness for purpose. Put another way, from the perspective of activity theory [8], the activity of design must have an intended outcome, and evaluating all other supporting actors, rules, tools, *etc.*, must be done by considering them as being oriented toward the object that produces the desired outcome. Hence, the course focuses on helping students learn a design process that includes identifying and expressing the purpose of a design and evaluating different activity facets that can support the process. We apply this to both user-facing design as well as internal software architecture. The students read Norman's classic *The Design of Everyday Things* [20] and consider how its topics impact their everyday digital and analog experiences, which allows for discussion of how design theory was used to produce those experiences. The students also learn to use and compare different design principles and models, including Norman's design principles, Gestalt principles of design, elementary accessibility heuristics, and heuristic evaluations for usability [18].

This perspective on design provides a counterpoint to common student experiences. They are often told what to build by an instructor, and so they build for the instructor within some set of given constraints. Learning that design is about fitness for purpose is revolutionary to students who, until then, have been doing some mix of bad design and non-design. They begin to see that design is always about tension. This forms a theme for the course, an essential question [16] that binds different pedagogic activities together. In order to understand that design is about trade-offs, students must first understand their goals and how to know if they have met them, otherwise there can be nothing traded away—no evaluation. Students benefit from understanding that design is inherently a conversation between reasonable alternatives. That is, a good designer considers that different users have different needs.

3 The OpenMind Platform

The OpenMind Platform provides eight free online lessons, each taking about twenty minutes to complete. The titles of the eight lessons are indicative of their content: explore the inner workings of the mind; uncover the roots of our differences; cultivate intellectual humility; welcome diverse perspectives; explore other worldviews; challenge the culture of contempt; manage emotions during difficult conversations; and master difficult conversations. In Fall 2020, there were five steps rather than eight lessons, the names of which are given later in Table 1. This report focuses on OpenMind as it was at that time. Both the eight-lesson and five-step forms follow common patterns for Web-based interactive content: short text presentations followed by reflective questions, branching options, and automated feedback.

The OpenMind Platform avoids ideological or partisan frameworks in favor of empirical ones. It describes tensions between the political "right" and "left" as phenomena worthy of consideration without praising nor condemning either. Various social sciences are used to support their analytical lens, especially moral foundations theory [10]. These are combined with humanistic insights from philosophy and various world religions. For example, it advocates for the triumph of humility and generosity of spirit over the culture of contempt, which position can be found in many faith traditions.

Readers familiar with the work of social psychologist Jonathan Haidt will recognize his influence. Haidt is the Co-Founder and Chairman of OpenMind, and the platform echoes ideas that can be found in in his public lectures, his popular book *The Righteous Mind* [12], and his scholarship. Moral foundations theory in particular underpins the theses of Haidt's subsequent work with Greg Lukianoff [15]. They argue that modern educational practices have inculcated youth into believing three "great untruths" of Fragility, of Emotional Reasoning, and of Us vs. Them. Each can be pithily summarized as "What does not kill me makes me weaker," "Trust your feelings," and "Life is a battle between good people and evil people," respectively. They call these "untruths" because they stand athwart both social science findings and classical values. The first is contrary to humans' antifragility, that many of our physical and psychological systems become stronger under stress. The second is contrary to what we know empirically about psychological phenomena such as motivated reasoning and confirmation bias, that our feelings often distract us from finding truth or compromise. The third appeals to the worst of our tribal instincts, related to the previous point, which leads us to caricature those who disagree with us.

The comparison of the great untruths to global classical values is reminiscent of the appeal to the *Tao* in Lewis' [14], "The Abolition of Man." Lewis considers an amalgam of global traditions where practical teachings intersect regardless of theological differences. Despite Lewis' well-known religious con-

victions, his appeal to the Tao is secular, looking at the shared wisdom of culture rather than relying on his experience or opinion of divine revelation. This pattern continues in The OpenMind Platform, which makes no claims about the absolute authority of any particular traditional worldview but draws upon them as points of reference, inspiration, and validation.

Systematic evaluations of The OpenMind Platform are underway. The OpenMind blog cites the as-yet unpublished results of their first randomized control trial [21]. A recent conference presentation discussed the positive preliminary findings [6], and there is at least one major, cross-institutional research study on the platform's efficacy currently being conducted.

4 Integration

The HCI class was offered in an asynchronous online mode in a fifteen-week semester. Each week was presented as a thematic module containing a variety of activities. Through these, students created and shared various works to represent their understanding, following the essential characteristics of studio-based learning despite being online and asynchronous [17]. Student-created works included reflections, essays, programming projects, and design artifacts. The first nine weeks explored a variety of themes in design-focused HCI, and it was during this period that OpenMind was used. The remaining six weeks of the semester were spent on a final project whose primary deliverable was a comprehensive report documenting the design, development, and analysis of a Web-based software system, supported by a prototypical implementation. Structured integration with OpenMind took place during the first nine weeks as shown in Table 1. The complete published plans for the course are published at `https://www.cs.bsu.edu/~pvgestwicki/courses/cs445Fa20` under a Creative Commons Attribution Share-Alike license.

The first week of the course provided an introduction to design following the first chapter in Norman [20]. A theme of the week was that good design solves a problem—usually for someone else. Part of the week's activity included completing the first OpenMind component. This was framed in the course plan as part of a dedicated effort to learn how to disagree productively, a reaction to the political and even scholarly tribalism.

The first OpenMind assignment explained motivated reasoning and confirmation bias. Their assignment, then, was to reflect on a previous course project and consider whether these two phenomena were identifiable in them. Responses were shared online via discussion board, establishing a pattern of sharing that would be followed during the semester. Students earned credit by reading and responding to each others posts.

In the second week, students learned about the seven stages of action as

Table 1: Alignment of Semester Weeks, Themes, and OpenMind Steps

Week	Weekly Theme	OpenMind Step
1	Introduction to Design	Explore the irrational mind
2	Thinking, Acting, and Evaluating	Uncover the roots of our ideological differences
3	Memory and Mistakes	Cultivate intellectual humility
4	Principles and Processes	
5	Integrating Design Processes with Flutter	
6	Integrating Design Processes with Flutter	
7	Design Thinking and Heuristic Evaluation	Appreciate the value of diverse perspectives
8	Accessibility	Prepare for constructive disagreement
9	Visual Design and Personas	

well as design constraints as presented in chapters two and four of Norman [20]. These readings were paired with the second OpenMind step, which introduced moral foundations theory—that each person lives within a different moral matrix built from the shared foundations of care, fairness, loyalty, authority, sanctity, and liberty [10]. The theory explains that many disagreements stem from applications of different moral foundations. The corresponding course assignment then was to consider the relationship between moral foundations and Norman's design constraints, particularly what he calls "cultural constraints."

The third week covered Norman's classification scheme for human errors. As a response to this reading, the students were asked to share various errors they had made and to classify them within Norman's taxonomy. This was tied to the third step of OpenMind, which explains the value of intellectual humility—the belief that your own beliefs could be wrong [13]. The OpenMind Platform positions this idea in the context of supporting a growth mindset [7]. That is, intellectual humility is a key to a growth mindset, and a growth mindset is preferable to a fixed mindset. For the students' assignment, they were asked to consider how they presented the aforementioned errors—whether their own presentation was indicative of a growth or fixed mindset.

The fourth week of the semester introduced the double-diamond design model [3] and Gestalt principles of design [2]. There was not a clear mapping from the these to the next part of OpenMind. The fifth and sixth weeks also did not provide clear integration opportunities since they focused primarily on

technical issues of constructing graphical user-interfaces using Flutter[2]. Students built digital prototypes in Flutter during these two weeks, and these designs provided a subject for analysis in the following weeks.

The seventh week of the class returned to design-theoretic concepts, specifically, design thinking [4] and heuristic evaluation [19]. The principles of heuristic evaluation provided a good integration point for OpenMind Step 4, which builds upon the previous steps to make a social science argument in favor of the value of diverse perspectives. Its presentation emphasizes the contemporary danger of hyper-polarization. Students were asked to make explicit connections between design heuristics and diversity. They readily saw that different perspectives were valuable to both the design and evaluation processes.

The eighth week of the semester had two parts: an introduction to accessible design and the conclusion of the multi-week analysis that started with the Flutter tutorials. This week, the students also completed the fifth and final part of OpenMind, which brought all of the other threads together into a presentation of how to have constructive disagreements. The students then were asked to step back and consider the implications of the OpenMind Platform on their HCI studies.

5 Results and Discussion

The student response to the integration was overwhelmingly positive. Everyone said it was useful to their studies, and many provided unsolicited feedback that its impact reached well beyond the course. One student pointed out how his background in social science made him skeptical of the program, but that after having gone through it, he found nothing to be questionable nor overreaching. (Honestly, I had a similar experience in terms of my own expectations and reactions to having completed it.) Another student suggested that it would be even more valuable if given in high school to counteract the tribalism that arises there. The only ambivalent response was from a student who said they did not think they got anything from the OpenMind Platform in particular, but that they enjoyed the intellectual exercise of combining it with HCI themes.

Students' backgrounds provided important contexts for how they addressed the integration exercises. We can roughly divide the students into two groups: those who had significant community-engaged experiences through the two-semester capstone or work experiences, and those whose only experience working on a large, multi-week project was in a prerequisite course. That prerequisite course asks students to make software for an identified target audience, but it does not have the rigor of full-fledged requirements analysis and user testing. Students with limited exposure to "real" users tended to address the OpenMind

[2] `https://flutter.dev`

implications with predictable second-order ignorance. This contrasted against the other students, who made more piercing and nuanced observations about how the ideas of the OpenMind Platform arose in professional software development environments. Notably, students with more experience were more clearly able to articulate, in specifics rather than generalities, how users and developers (and managers!) are different.

Students regularly made unsolicited reference to what they learned in the OpenMind Platform when writing about other topics. A common reference was the importance of developing a growth mindset, particularly in the face of receiving peer and expert feedback on their designs. They also regularly referenced the elephant-and-rider metaphor, which observes that our emotional side is like an elephant and our analytical side is like its rider: the elephant provides the power, and while the rider can make a plan and see a path, it has limited control over the beast [11]. Another student adroitly recognized the connection between OpenMind's themes and those in Norman [20].

As mentioned earlier, recent enhancements to the OpenMind Platform bring it to eight lessons rather than five. I have no personal experience with this revised presentations, although I have given them as an optional source of credit for another course, and students who pursue this have given me only positive reports. An unfortunate consequence of the platform's expansion is that readers cannot simply apply the integration described here and on the public course site. Additional work would be required to integrate these eight lessons effectively into a design-focused HCI class.

6 Conclusions and Future Work

The OpenMind Platform provided a useful complement to undergraduate study of human-computer interaction. My experience has been that the OpenMind Platform improves students' understanding of course material, particularly around learning to work with different kinds of teammates and users. Future work could study more empirically how Computer Science majors and minors are affected by the experience.

The integration of OpenMind with my HCI course was a success. Many students lauded the integration and, knowing it was probationary, encouraged keeping it in the class. The students showed an open mind about using OpenMind, reminiscent of Chesterton [5], "The object of opening the mind, as of opening the mouth, is to shut it again on something solid." This "something solid" presented by the OpenMind Platform is an empirically-supported understanding of the human condition that encourages students to work well with others in a diverse society.

The positive experience with OpenMind suggests that it is worth investi-

gating whether it can be integrated at other points in the curriculum. At my institution, HCI is an elective that is taken by a minority of students, but I suggest that OpenMind is more broadly useful. I have recently added it as an optional credited experience in a required sophomore-level programming class [9]. Several students took the opportunity, and all who did reported a positive experience.

7 Acknowledgments

I am grateful to Lauren Alpert Maurer and the staff at The OpenMind Platform for their assistance in providing information and references for this report.

References

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. *Computer Science Curricula 2013*. Dec. 2013.

[2] Rudolf Arnheim. *Art and Visual Perception: The Psychology of the Creative Eye*. Berkeley: University of California Press, 1954.

[3] British Design Council. *Framework for Innovation*. `https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond`, accessed August 11, 2021. 2019.

[4] Tim Brown. "Design thinking". In: *Harvard business review* 86.6 (2008), p. 84.

[5] Gilbert Keith Chesterton. *The Autobiography of G. K. Chesterton*. London: Sheet and Ward, 1936.

[6] M. C. Dieffenbach et al. "OpenMind: A scalable online intervention to depolarize campuses and communities". Conference presentation at Virtual SPSP 2021 Convention. Feb. 2021.

[7] Carol S. Dweck. *Mindset: The New Psychology of Success*. New York: Random House, 2006.

[8] Yrjö Engeström. *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. second. Cambridge, Mass.: Cambridge University Press, 2014. DOI: `10.1017/CBO9781139814744`.

[9] Paul Gestwicki. "Design and Evaluation of an Undergraduate Course on Software Development Practices". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 221–226.

[10] Jesse Graham et al. "Moral Foundations Theory: The Pragmatic Validity of Moral Pluralism". In: *Advances in Experimental Social Psychology*. Ed. by Patricia Devine and Ashby Plant. Vol. 47. Cambridge, Mass.: Academic Press, 2013, pp. 55–130.

[11] Jonathan Haidt. *The Happiness Hypothesis: Finding Modern Truth in Ancient Wisdom*. New York: Basic Books, 2006.

[12] Jonathan Haidt. *The Righteous Mind: Why Good People are Divided by Politics and Religion*. New York: Pantheon, 2012.

[13] Mark R Leary et al. "Cognitive and Interpersonal Features of Intellectual Humility". In: *Personality & social psychology bulletin : journal of the society for personality and social psychology* 43.6 (2017), pp. 793–813. ISSN: 0146-1672.

[14] Clive Staples Lewis. *The Abolition of Man*. New York: Macmillan, 1947.

[15] Greg Lukianoff and Jonathan Haidt. *The Coddling of the American Mind*. New York: Penguin Press, 2018.

[16] J. McTighe and G. Wiggins. *Essential Questions: Opening Doors to Student Understanding*. Alexandria, VA: ASCD, 2013. ISBN: 9781416615705.

[17] N. Hari Narayanan et al. "Transforming the CS Classroom with Studio-Based Learning". In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 165–166.

[18] Jacob Nielsen. "Heuristic Evaluation". In: *Usability Inspection Methods*. Ed. by Jakob Nielsen and Robert L. Mack. New York: John Wiley & Sons, 1994.

[19] Jakob Nielsen and Rolf Molich. "Heuristic Evaluation of User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '90. Seattle, Washington, USA: Association for Computing Machinery, 1990, pp. 249–256.

[20] Don Norman. *The Design of Everyday Things*. Revised and Expanded. New York: Basic Books, 2013.

[21] OpenMind. *Randomized Controlled Trial Demonstrates OpenMind's Effectiveness among Adult Learners*. `https://openmindplatform.org/blog/randomized-controlled-trial-effectiveness-adult-learners/`. n.d.

[22] M. Stiksma. *Understanding the Campus Expression Climate: Fall 2020*. Heterodox Academy. 2021.

Strategies for Equitable Participation in an Introductory Computer Science Course*

Meredith Moore and Timothy Urness
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50131
meredith.moore@drake.edu, timothy.urness@drake.edu

Abstract

In this paper we describe techniques intentionally designed to promote inclusivity and equity in an introduction to computer science course. We describe approaches for using randomly-drawn name cards as an alternative to cold-calling students for participation in class. We also discuss using an online polling technique that utilizes components of the Peer Instruction pedagogy to solicit low-stakes individual contributions. In each case, we motivate the "best practices" we have experienced and propose methods for making computer science classrooms more inclusive and equitable.

1 Introduction and Motivation

Computer Science is an incredibly influential discipline that provides an opportunity to develop technology to help solve both big and small problems in our world today. However, if the people developing the technology do not represent our communities or populations, then the solutions to the problems are unlikely to be representative, and as a consequence, will be limited in insights,

efficiency, effectiveness, and contain an unintentional bias [7] [6]. It is imperative that the classrooms where the foundations of CS education are introduced and established are welcoming and inclusive.

Efforts to include more women in computer science have been ongoing for decades [6, 1] [5,11]. While progress has been made, there is still a significant disparity in gender (77.7% male vs. 22.3% female) and ethnicity (39.8% white, 27.3% Asian, 15.6% Nonresident Alien, 9.6% Hispanic, 3.9% Black or African-American) of Bachelor's degrees awarded in Computer Science, Computer Engineering, or Information [1] [11]. Diversity in computer science is important. A diverse workforce that is representative of the people using technology is essential for solving hard problems that require perspective, creativity, and contributions from many. Furthermore, there is a tremendous demand for more computing professionals, and the industry cannot afford to be exclusive.

Many strategies have been shown to be effective in increasing the equity and inclusion of underrepresented students in computer science focused in the introductory (CS 1) course. These strategies include making the introductory course more welcoming [10] [8] and less intimidating [2, 8] [1, 9], making the assignments more meaningful [4, 11] [3, 10], building peer-tutoring pipelines [5] [4], and separating experienced CS 1 students from inexperienced CS 1 students to reduce intimidation, fear and imposter syndrome in novice students [3] [2].

In an effort to achieve inclusion and equity in the classroom, we would like to provide an environment where each student is not only provided with equal opportunities, but where each student is motivated to participate equally and each student feels like they belong in the computer science classroom. Our goal is to provide a classroom environment where every student knows each other, is known by others, and is both supported and supportive in achieving common goals of learning, developing foundational knowledge, and building meaningful software.

In order to create an environment of belonging in which students feel invited, equal, and included, we feel we need to be rigorously intentional regarding the following:

- Students feel safe to participate in the classroom
- Students do not feel ignored, neglected or left behind
- If a student is falling behind, there is a proactive approach of identifying them and reaching out and assessing what additional resources may be needed

Towards this end, we describe two different approaches we have adopted in CS 1 that are designed to promote inclusion and equity, and provide a scaffold for early identification of struggling students. The first approach describes the use of cards with each student's names on them which are used to help

ensure every student in the class has an equal opportunity to contribute. The second approach is an adapted format of Peer Instruction using PollEverywhere (`https://www.polleverywhere.com/`) to motivate all students to participate and provides an assessment mechanism that can provide early detection for struggling students.

2 Participation Cards

One of the most difficult tasks for a professor is to present course material at an appropriate pace for the students taking the course. When asking a question of the class, it is often convenient to call upon the same small group of students – generally those who sit in the front, have raised their hands, and are making eye contact – to answer the question and move the presentation of the material along at a desired pace. Unfortunately, the approach of habitually calling on the most attentive students will not get a reasonable measure of how the average (or struggling) student is following the presentation or understanding the material. One possible solution to this challenge is to randomly cold-call on students. However, this practice can create a stressful and anxiety-inducing environment in which students feel pressured to always be "on" and could feel discriminated against or "picked-on" by the professor, causing a lack of trust with the professor and a lack of comfort in the classroom. As an alternative, we have found success in creating an inclusive environment by using a deck of cards in which students have written their name.

2.1 Implementation

At the beginning of the semester, a deck of blank cards (either index cards or blank playing cards) can be distributed to the students. Each student writes their name on the card with any helpful phonetic spelling as needed (or other relevant information: year, major, etc). This deck of cards is then used throughout the class when a question is poised by the professor. The top card is drawn, the student is called upon, then the card is discarded or placed at the bottom of the deck.

We found that being explicit about the "why" of using the cards – explaining that they are a tool to invite more equity into the classroom – is an important key to reducing the anxiety around having students' names called out. We also found that it is also important to take extra care in explaining to the students that the goal is for all to learn, and that there is absolutely no shame in saying "pass", "I don't know", "can you repeat the question", or even using a life-line by asking the the next card to be drawn for another student to help in answering the question.

In practice, we have found it best to shuffle the cards, draw the top card, and have the corresponding student be the "card bearer" for the rest of the class period. Thus, when a question is posed to the class, the "card bearer" student takes the top card and announces the student who will have the first opportunity to respond. This helps take the burden off of the professor of being the "bad guy" and also helps all of the students get to know each other better in the class. Having students say each other's names, and look around the room to make eye-contact with the person whose name they just said is a great way to give students an incentive towards getting to know who is in their class. This worked particularly well for our use case with relatively small classes (30 students).

Creative uses of these cards include using them to take attendance by going through the deck, calling students names and separating into "present" and "absent" piles, dealing out cards to create groups, as well as using them to create a random seating chart. In future classes, we plan to add an element of gamification by adding a few "reverse" cards where the professor will answer their own question, as well as a "draw two" card where the question is posted to the next two students selected in the deck who can then collaborate to produce an answer.

2.2 Advantages

Using this approach has several advantages. Students know it is fair (and equitable). Students know that being called upon isn't done out of a professor's spite but just the "luck of the draw." Our goal is to create an atmosphere where students adopt the attitude of "we are here to learn together." The cards help provide an inclusive practice and establish more equity as each student has an equal chance of being called upon.

Student feedback on Participation Cards (included with their permission):

- "I appreciated them because I hate sitting through the long silence after a question is asked and only a handful of us are willing to speak up."
- "I liked that they gave students an option to pass, but also gave students a chance to answer questions without having to raise their hands and speak (which can be anxiety inducing)"
- "It gives everyone a chance to participate in class"
- "The class environment was really supportive of each other, I think that volunteering an answer was encouraged. I think that the cards with our names on them could've been used more"
- "It helped keep the class engaged and I thought they were good."

3 A Modified Approach to Peer Instruction

Peer Instruction is a well-documented pedagogical method that first asks students to individually respond to a multiple-choice-question posed in a classroom [9] [7]. After the initial question, students discuss in small groups, challenging each other to develop a consensus. Afterwards, the students answer the question again, oftentimes with improved results. Peer Instruction has been documented to be an effective approach to increase student performance on conceptual questions [9] [7]. An advantage of Peer Instruction is that it provides an opportunity for all students to participate simultaneously. As such, Peer Instruction helps prevent the non-equity practice of professors asking questions of the class and regularly calling on the same students or only students with raised hands.

Peer Instruction has many advantages; however, challenges in finding the right kinds of conceptual questions (one where not everyone initially agrees) and balancing the Peer Instruction pedagogy along with content delivery has motivated us to adopt a modified Peer Instruction approach to the CS 1 classroom. We believe this approach has been effective in the goal of making the classroom more inclusive and equitable.

3.1 Implementation

As an alternative to a hand-held "clicker" device that students would have to purchase, we require students to purchase a $15 semester subscription to PollEverywhere, which allows them to use their smartphones and computers to participate in the classroom Peer Instruction activities. The subscription also allows us to retain students' answers over the semester. While there is an option to poll students anonymously, we found it advantageous to utilize questions which attached the student's name to their response. The answers displayed in class never had any identifiable information; however, the professor could identify students' answers afterwards through the PollEverywhere account. One of the significant advantages to this method is that we can track attendance and participation.

In the standard Peer Instruction approach, students would collaborate after answering the question individually. In practice, we found that it could be redundant to have students complete the collaboration portion if the class was in agreement with their original answers. The collaboration component was most valuable when there was not a strong consensus in the initial vote. PollEverywhere provides the ability to display the results as they are entered or after all results have been submitted. Thus, if there was a strong consensus, it made the most sense to show the results and reinforce the correct answer with an explanation. However, if the students' answers were distributed across the different

options, the option to utilize the traditional Peer Instruction methodology by having students share their answers with their peers to see if they could come to a consensus was effective. This modified Peer Instruction approach is depicted in Figure 1.

We chose to use the answers in a low-stakes fashion where participation received full credit, regardless of correct or incorrect submissions. We feel that normalizing errors in CS 1 classes can help students not to get discouraged when they encounter bugs in their programs. Furthermore, when explaining this choice of low-stakes participation to students, we also took an opportunity to discuss the importance of having a growth mindset when learning computer science as well as reiterating that we are all in this classroom to learn and that one of the most effective ways to learn is to make mistakes.

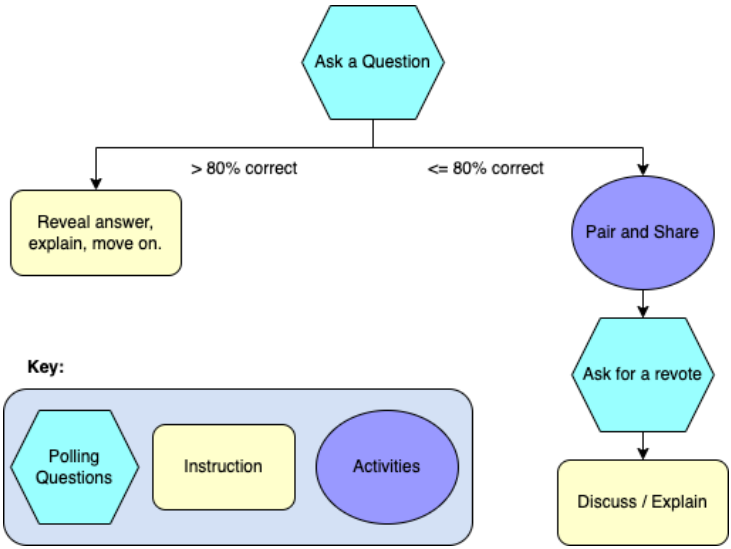


Figure 1: Flowchart for adapted Peer Instruction approach. Note that 80% accuracy is an estimate, and in practice, this threshold is malleable.

At a high-level, a typical class would begin with a low-stakes attendance question to get students responding and stimulate conversation (see table 1). We then briefly review the previous lecture's concepts and conduct a PollEverywhere review question. Next, we introduce new material, and then provide the students with a hands-on activity to practice the new topic. Once the students have had a chance to try out the new concept, we conduct another polling question using the modified Peer Instruction methodology. Finally, at the end of class, we ask students to reflect on how they feel about what they

have learned.

3.1.1 Attendance Questions

The attendance questions provide conversation starters for community building to take place in the few minutes before class has started. For most of these questions, we found having the answers displayed as they came in was a nice way for students to build off of each other's answers as well as start conversations. We begin a class with a simple, ice-breaker question that is primarily intended for attendance purposes. The goal of the attendance question is to set a non-threatening participation pool to facilitate participation throughout the class period.

Table 1: Example Attendance Questions

What's your favorite animal?
If you had a theme song, what would it be?
What is your favorite food?
What is your favorite hobby?
What was something that made you smile this weekend?
When you were a kid, what did you want to be when you grew up?
How are things going on Assignment 4?
Were you able to complete Lab 2?

Sometimes these attendance questions played a more administrative role – serving as a way to get a feel for how students were progressing with coursework. Questions like the examples in the last two bottom rows of Table 1 were effective in assessing how an assignment or lab was going for students. This also presents an opportunity for a follow-up question to help identify where students may be stumped.

3.1.2 Review Questions

After answering the attendance question(s), we would then move into a quick review of the material covered in the previous class session and provide an opportunity for students to test their understanding of the previous material with a review question. The review question provided an opportunity to assess students' understanding of the previous material and often facilitated conversations clarifying any misconceptions about the previous class's material.

3.1.3 New Concept Questions

We next would move on to introducing new concepts. Throughout the classroom session, we typically introduce a topic, provide examples, then give a short individual or small group exercise to give students an opportunity to practice the new concept in the class on their laptops computers. After the exercise is an opportune time for another polling question to reinforce the concept introduced.

3.1.4 Reflection Questions

At the end of the class period, we give a final poll asking students how they felt about their understanding of the material presented in the course using the 'clickable image' question type in PollEverywhere, as represented in Figure 2. This allowed students to rate, on a visual continuum, how confident they felt about the concepts talked about in class.

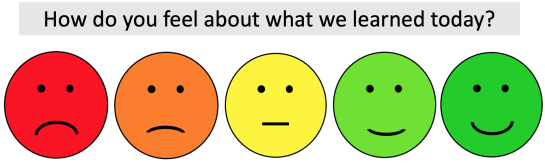


Figure 2: A representation of a "clickable image" question that was used to encourage students to reflect on what they learned.

Looking at answers from the reflection question after class can provide immediate feedback as to how a class period was received by students, as well as note students who may benefit from an individual contact and invitation to stop by office hours to discuss any difficulties. In our experience, this timely and proactive engagement can be instrumental in preventing a student who happens to be struggling with a particular concept from falling significantly behind in the course content. We feel this engagement can also facilitate and encourage future interactions with a struggling student and the professor. Finally, this end-of-class reflection question requires the student to actively reflect on their understanding of the material, which can also provide the impetus for them to request additional help on a topic that may not have made sense during the class period.

3.2 Advantages

We feel that the advantages of using a modified Peer Instruction approach are numerous. First, it builds equity in the classroom as the multiple choice polling

questions allow every student to have an input into the class.

The results from the surveys give professors an immediate metric as to the overall understanding of a topic. This is in contrast to a general "feel" for the understanding obtained by a professor reading the expressions of students in the classroom.

The PollEverywhere results allow us to track the responses for each student. Thus, a student's absence can be detected by the system and a proactive reach out (e.g. email, text message, or message to academic support staff) could be helpful in preventing them from falling behind. A dashboard from the results of the polls can be easily created to give more information about the overall health of the students in the course.

Lastly, the polling system keeps track of each student. Thus, it works well in small courses, but the effectiveness is not dependent on the number of students in the course as the polling software scales to larger classes.

Student Feedback on the use of PollEverywhere (included with their permission):

- "The [PollEverywhere] questions helped me understand the material we were covering while giving me examples/problems to think about and work through."
- "I felt like it kept me very engaged. It was a nice way to test my learning without being put on the spot."
- "PollEverywhere helped me to actually apply concepts and it helped me a lot in studying for exams!"
- "There were many ways it was used and it was a good way to see how people were doing and if there is anything that's not understood"
- "It is an easier way to practice questions in class"

4 Conclusion

Computer science is in high demand and is poised to help solve many different kinds of problems. The future of computing will be shaped by those people that feel welcomed into the discipline. Instructors in CS 1 courses have the potential to create environments that will foster a diverse and equitable learning environment that maximize the potential of individual students and increase the diversity of students that contribute to developing technology. In this paper we described techniques intentionally designed to promote inclusivity and equity employed in an introduction to computer science course. These include an approach for using randomly-drawn name cards as an alternative to cold-calling students for participation in class and using an online polling technique that utilizes components of the Peer Instruction pedagogy to solicit low-stakes individual contributions. It is our hope that adopting practices

like this will create a welcoming, inclusive classroom environment and help all students learn computer science.

References

[1] Computing Research Association. 2021 taulbee survey. `http://cra.org/resources/taulbee-survey/`.

[2] Pamela Burdman. To keep students in stem fields, let's weed out the weed-out math classes. *Scientific American*, 2022.

[3] James P. Cohoon and Luther A. Tychonievich. Analysis of a cs1 approach for attracting diverse and inexperienced students to computing majors. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 165–170, New York, NY, USA, 2011. ACM.

[4] Lucas Layman, Laurie Williams, and Kelli Slaten. Note to self: Make assignments meaningful. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 459–463, New York, NY, USA, 2007. ACM.

[5] Adamou Fode Made and Abeer Hasan. Creating a more equitable cs course through peer-tutoring. *J. Comput. Sci. Coll.*, 35(10):33–38, apr 2020.

[6] Jane Margolis and Allan Fisher. *Unlocking the clubhouse: Women in computing*. MIT press, 2002.

[7] Microsoft, Microsoft Philanthropies TEALS (Technology Education, and Literacy in Schools) partnered with the National Center for Women Information Technology (NCWIT). Guide to inclusive computer science education. `https://ncwit.org/resource/csedguide/`.

[8] Elaine Seymour and Anne-Barrie Hunter. *Talking about leaving revisited*. Springer, 2019.

[9] Beth Simon, Michael Kohanfars, Jeff Lee, Karen Tamayo, and Quintin Cutts. Experience report: Peer instruction in introductory computing. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 341–345, New York, NY, USA, 2010. ACM.

[10] Timothy Urness and Eric Manley. Building a thriving cs program at a small liberal arts college. *J. Comput. Sci. Coll.*, 26(5):268–274, may 2011.

[11] Jessica Zeitz and Karen Anewalt. Assignments to promote diversity and accessibility. *J. Comput. Sci. Coll.*, 34(3):18–19, jan 2019.

Reflective Curriculum Review for Liberal Arts Computing Programs

Conference Tutorial

Jakob Barnard[1], Grant Braught[2], Janet Davis[3], Amanda Holland-Minkley[4], David Reed[5], Karl Schmitt[6], Andrea Tartaro[7], James Teresco[8]

[1]University of Jamestown, Jamestown, ND 58405

Jakob.Barnard@uj.edu

[2]Dickinson College, Carlisle, PA 17013

braught@dickinson.edu

[3]Whitman College, Walla Walla, WA 99362

davisj@whitman.edu

[4]Washington & Jefferson College, Washington, PA 15317

ahollandminkley@washjeff.edu

[5]Creighton University, Omaha, NE 68178

DaveReed@creighton.edu

[6]Trinity Christian College, Palos Heights, IL 60463

Karl.Schmitt@trnty.edu

[7]Furman University, Greenville, SC 29690

andrea.tartaro@furman.edu

[8]Siena College, Loudonville, NY 12211

jteresco@siena.edu

Abstract

The ACM/IEEE-CS/AAAI curricula task force is currently developing an updated set of Computer Science Curricula guidelines, referred to as CS202X (since the release date is not yet determined). Information about the task force and preliminary drafts of the Knowledge Areas that will be included in the guidelines can be found online at http://csed.acm.org. To assist institutions in applying the new guidelines, CS202X will also publish a Curricular Practices Volume. This

volume will include an article by the SIGCSE Committee on Computing Education in Liberal Arts Colleges (SIGCSE-LAC Committee) that will focus on designing or revising CS curricula in liberal arts contexts. Liberal arts colleges, and smaller colleges in general, face unique challenges when designing curricula. Small faculty sizes, limits on the number of courses that can be required for a major and the need for flexibility in student programs of study constrain designs. However, these environments also provide the opportunity to craft distinctive curricula fitted to institutional mission, departmental strengths, locale, student populations and unique academic experiences. These challenges and opportunities, combined with the size of prior curricular recommendations, have often forced smaller programs to assess trade-offs between achieving full coverage of curricular recommendations and their other priorities.

The SIGCSE-LAC Committee has heard from many faculty that their institutional and departmental contexts have indeed complicated the adoption of prior curricular guidelines. While the CS2013 and upcoming CS202X recommendations provide some flexibility for curriculum designers by dividing content into core and supplemental categories, smaller colleges still face challenges selecting content and packaging it into coherent curricula. To assist in this process, the committee is developing guidance for effectively integrating CS202X as a part of the design, evaluation and revision of computer science and related programs in the liberal arts. This guidance will encourage faculty to reflect on their programs and the role of CS202X, beginning with their institutional and departmental priorities, opportunities and constraints. Ultimately, this guidance will be presented in the committee's article in the CS202X Curricular Practices volume.

This session will open with an overview and brief discussion of the current CS202X draft. Participants will then begin working through a preliminary version of the committees' reflective assessment process. This process is framed by a series of scaffolding questions that begin from institutional and departmental missions, identities, contexts, priorities, initiatives, opportunities, and constraints. From there, participants will be led to identify design principles for guiding their curricular choices including the CS202X recommendations. Participants will leave the session with a better understanding of how CS202X can impact their programs and a jumpstart on the reflective assessment process. Feedback on the process and this session are welcome and will be used to refine the committee's guidance prior to its publication in the CS202X Curricular Practices volume.

Presenter Biography

One of the eight co-authors of this session plans to serve as presenter. **Andrea Tartaro** is an Associate Professor of Computer Science at Furman University. Her computer science education research focuses on the intersections and reciprocal contributions of computer science and the liberal arts, with a focus on broadening participation. She is a member of the SIGCSE-LAC Committee, and has published and presented in venues including the CCSC and the SIGCSE Technical Symposium.

Other Author Biographies

Jakob Barnard is Chair and Assistant Professor of Computer Science & Technology at the University of Jamestown. He is a member of the SIGCSE-LAC Committee and his research involves how curricula has been integrated into Liberal Arts Technology programs. **Grant Braught** is a Professor of Computer Science at Dickinson College. He is a facilitating member of the SIGCSE-LAC Committee, has organized committee events focused on curricula and has published widely on issues related to CS education, particularly within the liberal arts. **Janet Davis** is Microsoft Chair and Associate Professor of Computer Science at Whitman College, where she serves as the department's founding chair. She co-organized SIGCSE pre-symposium events in 2020 and 2021 on behalf of the SIGCSE-LAC Committee. **Amanda Holland-Minkley** is Chair and Professor of Computing and Information Studies at Washington & Jefferson College. Her research explores novel applications of problem-based pedagogies to CS education at the course and curricular level. She is a facilitating member of the SIGCSE-LAC Committee. **David Reed** is a Professor of Computer Science and Chair of the Department of Computer Science, Design & Journalism at Creighton University. He has published widely in CS education, including the text *A Balanced Introduction to Computer Science*, and served on the CS2013 Computer Science Curricula Task Force. **Karl Schmitt** is Chair and Assistant Professor of Computing and Data Analytics at Trinity Christian College. He has served on the ACM Data Science Task Force and various Computing, Technology, Mathematics Education related committees for the MAA and ASA. His interests explore data science education, and interdisciplinary education between computing, mathematics, data, and other fields. **Jim Teresco** is a Professor of Computer Science at Siena College. He has been involved in CCSC Northeastern for almost 20 years and currently serves as regional board chair, and has been involved with the SIGCSE-LAC Committee for 3 years. His research involves map-based algorithm visualization.

Interdisciplinary Project-Driven Learning in Game Design and Development*

Panel Discussion

Seth Berrier, Karl Koehle, Kimberly Long Loken,
Michael Tetzlaff, Tyler Thomas
Information Sciences and Technology
University of Wisconsin – Stout

1 Summary

This panel will discuss the game design and development (GDD) program at the University of Wisconsin – Stout, with an emphasis on its use of project-based learning in an interdisciplinary setting, and the student-led use of Agile practices like sprints, stand-up meetings, reviews and retrospectives. The panel will briefly discuss the history of the program at Stout before diving into the curriculum used, from the foundational course taken by all first-year students to the senior-year capstone experience. It is intended that the takeaways from the panel will have applications to computer science education in general, particularly the focus on interdisciplinary student projects.

2 Introduction

UW-Stout is nationally recognized for its strong game design program, ranked by the Princeton Review as 6th in the U.S. among public universities[1]. Founded in 2008 by Dr. Diane Christie as a concentration within Applied Math and Computer Science and expanded in 2011 by Dave Beck with a parallel BFA program for artists, Stout currently offers two undergraduate degrees related to game design: a B.S. in Computer Science with a concentration in Game Design and Development, and a B.F.A. in Game Design and Development – Art. Stout also has an M.F.A. program in Design, comprised of students

[1] `https://www.princetonreview.com/college-rankings?rankings=top-50-game-design-ugrad`

Figure 1: Students designing a tabletop game in GDD 100.

from varied disciplines with interest in cross-collaboration and independent research. The undergraduate program is anchored by a "GDD course" in each of a student's four years, which emphasize the interdisciplinary nature of game development. In these project-driven courses, artists and programmers work together to create and release a game; for advanced courses, students partner with other institutions (Michigan Tech or Berklee College of Music) for sound effects and music. The foundational (first year) GDD course is taught by Art Design faculty, the intermediate courses are typically led by Computer Science faculty, and the capstone is team taught to leverage unique expertise. Projects increase in length, complexity, and number of team members each year. Games are presented at a "Stout Game Expo" at the end of each semester where students from all GDD courses present their playable games to the public. In GDD courses, students learn crucial career skills, working with those outside their discipline in an Agile development context. Teams in each class are expected to produce design documents, which offer an opportunity to write professional documentation, define project standards, and develop language for team communication. Learning objectives in GDD courses are met primarily through experiential learning; most GDD courses have relatively little lecture and more hands-on time for the students. Each GDD course is four credits but has six classroom hours scheduled per week, similar to a lab. Students use Agile development practices with a rhythm of stand-up meetings every day the class meets, and sprint review meetings at the end of each sprint (typically every two weeks) followed by team retrospectives.

Figure 2: Students present a capstone game at Stout Game Expo.

3 GDD 100: Intro to game design

The foundational GDD course is fundamentally a design thinking course taught through the lens of games. Students become adept with rapid prototyping, whether crafting tabletop games or bodystorming variations on sports. The exploration of puzzle design becomes an entry point into heuristics and problem-solving processes. Frequent playtesting is recorded and iterated upon via flowcharts for rules and structure and hand-drawn data for balancing mechanics and variables. Students are also introduced to principles of accessibility and user interface, industry roles, notable ludologists, and non-commercial game genres.

4 GDD 200: Video game development

The sophomore year GDD course is most students' first opportunity to create a computer game. Students transfer their knowledge of game design theory from GDD 100 to the digital realm to create a 2D video game for web or mobile devices. For programmers, this class serves as a chance to apply their knowledge from foundational CS classes to an applied project, while for artists, it allows them to practice working in a digital art pipeline from asset creation to engine implementation. This is typically students' first experience with Agile practices like stand-up meetings and sprints. The placement of this course in the sophomore year gives students an early chance to work on a major team project to deliver a digital product, and the games produced can be excellent resume and portfolio material as students look for their first internship.

5 GDD 325: 2D game design and development

The third course in the GDD sequence focuses on project management and team skills, with an emphasis on things like revision control, team communication, and Agile best practices. GDD 325 is the first class where students work with external collaborators for music and sound effects. In the most recent offering of the course, students were required to learn a new game engine on their own (most chose Godot), teaching them how to read documentation and possibly learn a new programming language without depending on traditional classroom instruction.

6 Intermediate discipline-specific support classes

Several discipline-specific classes have been developed over the years to support the project-oriented GDD courses. For artists, DES225 (Pixel Vector Art) teaches skills, tools, and workflows needed to create 2D video game assets and implement interactivity in a game engine. This course lays an early foundation of skills that can be applied on team projects. Similarly, DES350 (Game Art and Engines) fills a pre-capstone need for B.F.A. students, creating depth in the Maya-to-game-engine pipeline, exploring both Unity and Unreal as well as principles of level design.

For programmers, CS326 (Programming in Game Engines) helps students apply programming concepts to game engines like Unity 3D and Unreal. Students learn the structure of these engines, algorithms and programming patterns that are common in game design, and develop the most important skills they will need in the capstone sequence. CS 343 (Computer Graphics) has also evolved in recent years from a fairly traditional OpenGL course to one focused on practical shader programming, using the openFrameworks C++ library to accelerate low-level graphics tasks.

7 GDD 450 – 451: Capstone

GDD 450 and GDD 451 form a two-semester sequence that is the culmination of the game design and development program at Stout. Students work in larger teams (compared to previous GDD courses) over the course of an academic year to make a 3D video game. During capstone year, students have access to a senior design space, a large room with tables and rolling whiteboards that facilitates design conversations better than a computer lab. Students are fully responsible for managing their own projects and assign team members to project management roles like running stand-up meetings, taking meeting notes, etc. There is an increased focus on critique, with several major reviews

each semester where students present to a panel of GDD faculty (not just the students' direct instructor(s)) and receive feedback to help the students both refine their own skills and produce a better final product at the end of the year.

Figure 3: A faculty critique for a GDD capstone project, held in the senior design space.

8 Conclusion

Game design is an inherently interdisciplinary field. As GDD students progress through their individual programs, they come together annually for group projects which allow them to demonstrate their skills. These small teams begin to resemble independent game studios, continuing to grow in size and scope by introducing more focused skills and stakeholders from both inside and outside the institution. Students are allowed to embrace their creative vision, set community standards, and encourage each other towards the final deliverable, while practicing and improving essential career skills that make them highly sought after upon graduation, both within and beyond the game development industry.

9 Biographies

Seth Berrier is an Associate Professor of Computer Science at UW-Stout, with research interests that include property capture via photogrammetry applied to game design. Dr. Berrier teaches a variety of advanced computer science and game design courses.

Karl Koehl is a Lecturer of Game Design and Development – Art at UW-Stout. He has over a decade of industry experience in 3D Animation and video production and is currently pursuing his MFA in Design. Classes taught include 2D and 3D Game Design, and Pixel Vector Art.

Kimberly Long Loken is an Associate Professor and Program Director of Game Design and Development – Art at UW-Stout, with professional interests including ecosystem services and games for change. She teaches a variety of design and game development courses.

Michael Tetzlaff is an Assistant Professor of Computer Science at UW-Stout. Dr. Tetzlaff's research background is in interactive computer graphics, and he teaches a range of computer science and upper-level game design courses.

Tyler Thomas is an Assistant Professor of Computer Science at UW-Stout. Dr. Thomas's research has focused on the intersection of cybersecurity and software engineering. He currently teaches Video Game Development, Cybersecurity, and a range of other courses.

Flutter: n Platforms, 1 Codebase, 0 Problems*

Conference Workshop

Michael P. Rogers
Department of Computer Science
University of Wisconsin Oshkosh
Oshkosh, WI 54901
mprogers@mac.com

Bill Siever
McKelvey School of Engineering
Washington University in St. Louis
St. Louis, MO 63130
bsiever@wustl.edu

This workshop will introduce participants to app development using Flutter[1], an open-source, cross-platform development kit from Google that allows developers to create apps for iOS, Android, web, and desktop (macOS, Windows, Linux), from a single codebase.

As a new product (first released in 2017), unencumbered by the need to support legacy code, Flutter's designers were free to choose from and enhance the best features of existing frameworks.The result is an SDK, based on a declarative UI paradigm, that is growing in popularity in industry. Apps are written in Dart, a 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 instructor or 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 software engineering / capstone class, a mobile computing class, or any other situation where students need to develop for more than one platform.

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 a mobile class.

References

[1] Google. *Flutter - Build apps for any screen.* en. `https://flutter.dev/`. (Visited on 07/10/2022).

Summary Words and in the News*

Nifty Assignment

David L. Largent
Department of Computer Science
Ball State University, Muncie, IN 47306
dllargent@bsu.edu

I often have learners who are not prepared. They arrive at class having not read or thought about the course material. This creates a significant challenge for them when the class session is discussion-based, rather than lecture. To increase learner preparedness, I started using two assignments a few semesters ago: Summary Words, and In the News. Summary Words (Words) requires preparation before class, and In the News (News) requires application of the course material.

Words is a two-part assignment: add words to a provided Google document, and then discuss with their peers in a discussion board. The learner reads the assigned material, and then records in the Google document what they believe to be the most important topics by listing a combined total of five individual words for all topics. By condensing the assigned reading to five words, they must identify the essence of the material. I have them add their words to a Google document so we can view a single combined list of words during our class discussion.

News is a two-part assignment: find, post, and discuss article information in a discussion board, and then discuss the articles with their peers in the discussion board. After we discuss the material in class, the learners find and briefly report on an article (published within the last eighteen months) related to the material. In their post, they describe how it impacts society, what social or legal issues could arise, and if it agrees or disagrees with the readings. By finding and relating an article to the assigned readings, they are demonstrating their understanding of the material.

Both assignments provide the learners an opportunity to interact with and learn from their peers. This discussion may lead to a better understanding of the material or may raise questions for them to consider.

Computer Network Between Two Departments Using Cisco Packet Tracer*

Nifty Assignment

Imad Al Saeed
Computer Science Department
Saint Xavier University, Chicago, IL 60655
alsaeed@sxu.edu

This assignment is for a course on Introduction to Networks for students with no prior knowledge with computer network. The course instructor explains the LANs and WANs concepts as a solution for ensuring communication among two or more LANs over the internet. These are the main requirements of computer networking and how they can share resources between them. Finally, the instructor explains different types of networking protocols with respect to the Routing Information Protocol used to ensure the communication between two or more computer networks. This assignment requires using a special software called Cisco Packet Tracer where students can download for free from the Cisco Academy site (Netacad.net) and install it on their computers. Students should the newest version of the software to do this assignment. This process should be done in three steps:

Step 1: Requirements A student should use the Cisco Packet tracer to build their first LAN, which consists of two Routers of 1841 type, four Switches of 2960 type, eight computers (desktops or laptops), and straight forward ethernet cables.

Step 2: Building the network Connect each two computers to a switch and then connect each switch to each interface on the router. Repeat the same process to build the second network. After that connect the two routers through the serial interface together.

Step 3: Configure the networks Students should use the Command Line Interface to assign the IP address to each interface (Fast ethernets and serial interfaces) on both routers. Then, they should configure the RIP protocol to ensure the communications between the routers using the router graphic user interface by assigning the network IP for each interface on both routers. Finally, students should ping the computers located on different interfaces to ensure the communication between the computer networks.

Computer Disassemble and Rebuild Days*

Nifty Assignment

James Roll
Department of Computer Science
University of Findlay, Findlay, OH 45840
rollj@findlay.edu

3C Computer Repair is a student run computer repair business at the University of Findlay. Students can work at 3C Computer Repair to gain experience working for a small business while working towards certifications, including TestOut PC Pro and CompTIA A+ in a classroom setting. The classroom side of the 3C computer repair experience gives students knowledge that is useful in the business, and the computer repair business gives students hands on experience with tasks that are covered in the certification exams. 3C Computer Repairs sees a wide variety of issues, include laptop screen and keyboard replacements, primary storage transferred and replacements, and malware removal and recovery. However, there are many important hardware issues covered in certification exams that do not frequently come up in 3C Computer Repair jobs. Computer disassemble and rebuild days were designed to give students an opportunity to get some hands on experience in these other areas.

3C Computer Repairs owns a variety of old PCs which are used for the activity. The activity is split over two class periods, one for students to disassemble the PCs down into their component parts and properly store them in static shielding bags, and another to rebuild the PCs and test them for functionality. Students are divided into small groups to work on the projects. This gives students hands on experience seating and removing RAM and processors from a motherboard, placing a motherboard and power supply in a computer case, and proper internal cable management. Many other learning opportunities are emergent from the activity as well. If the PCs do not function after being rebuilt, the students will need to troubleshoot to determine why. The activity is enjoyed by the students and provides them with a unique learning opportunity.

Buried Wireless Sensor Node Based on Internet Wi-Fi and Bluetooth technology for Precision Agriculture

Work In Progress

Ahmed A. Elmagrous

Mathematics, Statistics, and Computer Science Department
University of Wisconsin-Stout
Menomonie, WI 54751
elmagrousa@uwstout.edu

Abstract

According to the United States Department of Agriculture (USDA), the average farm size has increased from about 600 acres in the early 1980s to at least 1100 acres today, and there are many farms 510 times larger than that [1]. Adoption of sensors in agriculture plays a vital role in enhancing the overall production of crops and reduced environmental impact [2, 3]. Wireless Sensor Network (WSN) is a powerful technology for monitoring various parameters of farming lands, such as humidity, soil moisture, climatic condition, water quality, pests, weeds, and livestock [4]. WSN also can easily collect and deliver real-time information on the field and crop conditions that enables growers to improve crop production and minimize input costs. Installing numerous WS nodes (Super-Node station) lead to a precise site-specific management model that supports farming decision systems by collecting more information. However, installing number of the super-nodes will sharply increase the cost of the network. Moreover, as the number of the super-node stations increased in the field, the traffic of the farming machines will be difficult.

This work presents a buried wireless sensor node where the farmer can control the WSN from his/her mobile via the internet Wi-Fi or the Bluetooth technology. The farmer can send all the nodes to hide underground when the machine moves in the field. After the machine finish its work, the farmer can call all the nodes to show up again to continue collecting and sending the data to the super-node.

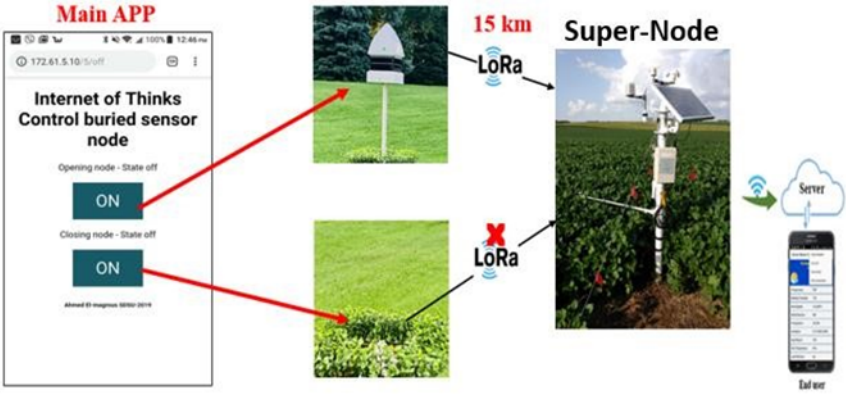


Figure 1: Buried Wireless Sensor Node.

References

[1]P. K. James M. MacDonald, and Robert A. Hoppe, "Farm Size and the Organization of U.S. Crop Farming," United States Department of Agriculture, August 2013. [Online]. Available: https://www.ers.usda.gov/webdocs/publications/45108/39359_err152.pdf

[2]B. Liu, "Wireless Sensor Network Applications in Precision Agriculture," Journal of Agricultural Systems, Technology, and Management, vol. 29, pp. 25-37, 2018.

[3]J. Polo, G. Hornero, C. Duijneveld, A. Garcia, and O. Casas, "Design of a low-cost wireless sensor network with UAV mobile node for agricultural applications," Computers and electronics in agriculture, vol. 119, pp. 19-32, 2015.

[4]A. Z. Abbasi, N. Islam, and Z. A. Shaikh, "A review of wireless sensors and networks' applications in agriculture," Computer Standards & Interfaces, vol. 36, no. 2, pp. 263-270, 2014.

Encouraging Student Voice with D, E, & I Based Online Communication Standards

Work In Progress

Kristi Hall

Clermont College
University of Cincinnati
Batavia, OH 45103
kristi.hall@uc.edu

Abstract

Over the years, studies have documented the importance of social interaction on student learning. However, there is a lack of information on how negative interaction impacts student learning. Given both the increase of online and hybrid learning, along with the increased toxicity in online communities on the Internet; the importance of diversity, equity and inclusion based online communication standards for online courses is imperative. As part of a faculty learning community at my institution, I completed a project where I involved my students in developing a new set of DEI-based "Online Communication Standards" to replace the outdated "Netiquette Rules" used by many online courses. This project has led to an ongoing research project to discover the impacts that negative online communication has on distance learning.

1 Introduction

There have been many studies conducted on the positive impact of social interaction on student learning, but very little on how negative interaction impacts the learning process (Xie, 2013). Common sense would suggest that negative interaction should negatively impact student learning. As an educator, I have seen how important community and a sense of belonging has on student success.

Anyone who uses the Internet has undoubtedly seen many examples of negative online communication. In the article Angry by design: toxic communication and technical architectures, Munn argues that decisions made during an online platform's design can promote negative online communication (Munn, 2020). If that, in fact, is true, logic dictates that we

should be designing our online learning environments to give our students opportunities to engage in positive online communications.

2 D, E, & I in Online Communication

It only takes a small amount of exposure to social media to see that much of the negative communication comes from the fact that people refuse to see that Internet users are a diverse group of people, with a wide array of diverse experiences and opinions on any given subject. The anonymity of the Internet contributes to this issue. It is much easier to shout down others and tell them they are wrong, because there are no consequences. It takes much more brain power to recognize that inclusion entails, thinking about the topic from another's perspective and then including them in the conversation. As in many areas of daily life, it is natural for us as humans to take the easy route.

I argue that the first step in encouraging student voice in our courses is to establish the framework that encourages our students to think before reacting. We must help our students to recognize the diversity of the people and ideas that they encounter in our classes, and then show them how to communicate and learn from each other without shutting down debate.

3 Online Communications Standards Project

During the academic year of 2021-2022, I was accepted into a faculty learning community at my university. For this community, I needed to develop a project that I would implement during the year. I am a web developer, with expertise in accessibility and I have worked hard to make sure my online courses are accessible, so the next logical step is to start making my courses more equitable and inclusive. After much thought, I decided to start this process by developing new diversity, equity and inclusion based online communication standards for my courses.

I have taught online for many years, and have seen an evolution in the field. With the growth of technology resources available for online learning, and with the increase in the number of students who have been using online communications on the Internet; it has become apparent that using traditional "Netiquette" for online course communications no longer benefits, nor facilitates, inclusive online communication in courses.

Netiquette was developed during the early days of the Internet. Most of those using the Internet during this time were researchers and systems administrators. The primary goal of Netiquette was to teach people to conserve computing resources because of the limitations of the technology of the time. Today, the technology we use is infinitely better than the technology used in the

early 1970s, which renders many Netiquette rules moot. The other component of this is the people: the people using the Internet now are not "computer people". George Margolin summed up the problem well when he said:

> All of a sudden you have this enormous influx of people who are not computer people. Try a 17-year-old druggie and tell me how he is going to behave on the Net. The etiquette of the Net will be no greater than the etiquette of the general population (Gornstein, 1999).

As I scroll through my social media, I see a lot of anecdotal evidence that proves he is correct.

Throughout my career, I have advocated for teaching digital citizenship skills across the curriculum. No matter what subjects our students are studying, technology will touch those subjects in many ways. So, I took this opportunity with my project to include my students. During the Spring Semester of 2022, I developed a lesson that I lead my students through in all of my courses. We started with a history lesson on Netiquette. This included how and why Netiquette came about, and how the changes in technology over the years have made most traditional Netiquette standards outdated. After that history lesson, we discussed online communication. I asked students:

- What their experiences had been with online communication.
- What their experiences had been with online communication in their courses.
- How they felt about online communication.
- What they would like to see in new online communication standards.

As I had this discussion with my students, I was quite surprised by the responses I was getting. It was quite evident that students had really been thinking about the topic of online communication without truly understanding the importance of what they were thinking. Some important topics that stood out during the discussion were:

- The topics of respect and kindness.
- Respectfully disagreeing with others.
- The impact of anonymity in online communication.
- The importance of proper writing skills.

At the end of the discussion, I began to compile the responses from my students and I used these responses to write the new "Online Communication Standards". Their responses fell into six general categories:

- Quality
- Professionalism

- Privacy
- Respect
- Tolerance
- Community Building

Once I had the standards written, I took them back to the students who contributed to them to get their input. The students were pleased with the results, and after some minor editing, I have the final version of the standards that I have moved forward with in my classes. I have made these available for use with a Creative Commons Attribution license on my web site at Online Communication Standards.

4 Conclusion

Now that the Online Communication Standards project for the faculty learning community is complete, the research component of the project will begin. I have planned a research study for the upcoming Fall Semester. All classes, regardless of teaching modality, at UC Clermont have an online classroom in the learning management system. For that reason, all students at UC Clermont will be asked to complete a survey about online communication. In this survey, they will be comparing the new standards to traditional Netiquette. The survey will gather their input on the new standards, and their experience with online communication in their courses to determine the impact of negative online communication on student learning.

References

[1]Gornstein, L. (1999). NETIQUETTE; Manners: Don't shout or waste bandwidth, or you'll get flamed. The Baltimore Sun. Retrieved February 3, 2022, from https://www.baltimoresun.com/news/bs-xpm-1999-01-25-9901250195-story.html

[2]Munn, L. (2020). Angry by design: Toxic communication and technical architectures. Humanities & Social Sciences Communications, 7(1), 1-11. https://doi.org/10.1057/s41599-020-00550-7

[3]Xie, K., Miller, N. C., & Allison, J. R. (2013). Toward a social conflict evolution model: Examining the adverse power of conflictual social interaction in online learning. Computers and Education, 63, 404-415. https://doi.org/10.1016/j.compedu.2013.01.003

Teaching Programming Paradigms Using CLIPS

Work In Progress

Ramachandra B. Abhyankar

Department of Mathematics and Computer Science
Indiana State University
Terre Haute, Indiana 47809
R.B.Abhyankar@indstate.edu

Abstract

CLIPS is an expert system shell, originally developed at NASA. In universities, it is often used in courses in Artificial Intelligence and Expert Systems and projects for building expert systems. I believe it can be used effectively in courses in Programming Languages to teach three programming paradigms: Rule-Based, Procedural, and Object-Oriented. CLIPS not only supports these programming paradigms, but also supports their integration.

Courses in programming languages that teach programming paradigms, often teach the paradigms in isolation. Prolog is often used to teach the Logic Programming paradigm. Haskell is often used to teach the Functional Programming Paradigm. There are several choices available to teach the Object-Oriented Paradigm: Smalltalk, Eiffel, Ruby, etc. Concurrent Programming is often taught using C, Java. However, Rule Based Programming is typically not taught in courses in Programming Languages as a programming paradigm.

Use of CLIPS in Programming Languages courses will not only support the teaching of the often-neglected Rule Based programming paradigm, but also teach the integration of this programming paradigm with the Object-Oriented and Procedural programming paradigms. CLIPS is a free download and comes with copious free documentation. Tutorials for CLIPS are also available in some books. CLIPS is a valuable resource for teaching multi-paradigm programming, which has been largely neglected by the academic community.

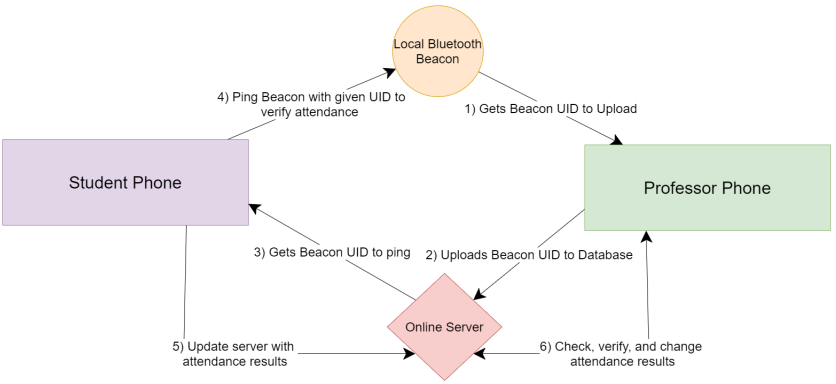
The presentation will show examples of CLIPS code that illustrate the different programming paradigms that it supports, and their integration.

Attendance Mobile Application*

Work In Progress

Zoltan Nahoczki, Matthew Fallon, Reed Mitchell
Department of Computer Science
University of Wisconsin Parkside
Kenosha, WI 53144
{nahoczki,fallon,mitcher1}@uwp.edu

When keeping attendance up to date it is difficult to keep it accurate. If a student shows up late an instructor might not notice and in result not mark them as late or attended. Likewise, if a student leaves class early this might also not get tracked. Attendance is an important part of a class environment, understanding which students are actively on time and which are not is important. It closely correlates with students who put in enough time for class. To solve this issue of tracking accurate attendance it is beneficial to use technology. The attendance mobile application would keep track of when a student attends class and leaves class. To do this an instructor would carry a bluetooth beacon with them which would be used to communicate with the student's mobile phones with the app installed. This would be a simple process. As long as the student's phone is within range of the beacon, we can assume that the student is still in class and can track attendance accordingly (see diagram below). Along with this, students would have a user interface where they can see their attendance for the classes using the application. Students would be able to see their percentage attended and a list with their attendance history. For registered instructors, they are able to create a course within the app, invite students to the course via an alphanumeric code, and manage the students. Managing students is an important piece to allow removing students and being able to manually set a student's attendance in case there are issues with bluetooth. In conclusion, attendance is an important piece to a course and keeping attendance data accurate is difficult. To solve this, a mobile application can track when a student attends and leaves class with visual data for both the professor and student.



Regular Pinging against the beacon throughout the class period verifies continued attendance.

Reviewers — 2022 CCSC Midwestern Conference

Dave Surma ..... Indiana University South Bend, South Bend, IN
David Bunde ..... Knox College, Galesburg, IL
David Largent ..... Ball State University, Muncie, IN
Deborah Hwang ..... University of Evansville, Evansville, IN
Edward Lindoo ..... Regis University, Denver, CO
Imad Al Saeed ..... Saint Xavier University, Orland Park, IL
Jean Mehta ..... Saint Xavier University, Orland Park, IL
James Vanderhyde ..... Saint Xavier University, Chicago, IL
Khaled Elzayyat ..... Salt Lake Community College, Salt Lake, UT
Michael Rogers ..... University of Wisconsin Oshkosh, Oshkosh, WI
Paul Gestwicki ..... Ball State University, Muncie, IN
Paul Talaga ..... University of Indianapolis, Indianapolis, IN