The Journal of Computing Sciences in Colleges

Papers of the 25th Annual CCSC Central Plains Conference

April 5th-6th, 2019
St. Charles Community College
Cottleville, MO

Papers of the 12th Annual CCSC Southwestern Conference

March 22nd-23rd, 2019
Stanford University
Stanford, CA

Volume 34, Number 4 **April 2019**

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. Printed in the USA. POSTMASTER: Send address changes to Susan Dean, CCSC Membership Secretary, 89 Stockton Ave, Walton, NY 13856.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

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

Jeff Lehman, President (2020), (260)359-4209, jlehman@huntington.edu, Mathematics and Computer Science Department, Huntington University, 2303 College Avenue, Huntington, IN 46750.

Karina Assiter, Vice President (2020), (802)387-7112, karinaassiter@landmark.edu.

Baochuan Lu, Publications Chair (2021), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Department of Computer and Information Sciences, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2020), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

Susan Dean, Membership Secretary (2019), Associate Treasurer, (607)865-4017, Associate Editor, susandean@frontier.com, UMUC Europe Ret, US Post: 89 Stockton Ave., Walton, NY 13856.

Judy Mullins, Central Plains Representative (2020), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, School of Computing and Engineering, 5110 Rockhill Road, 546 Flarsheim Hall, University of Missouri - Kansas City, Kansas City, MO 64110.

John Wright, Eastern Representative (2020), (814)641-3592, wrightj@juniata.edu, Juniata College, 1700 Moore Street, Brumbaugh Academic Center, Huntingdon, PA 16652.

David R. Naugler, Midsouth Representative(2019), (573) 651-2787, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

Lawrence D'Antonio, Northeastern Representative (2019), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

Cathy Bareiss, Midwest Representative (2020), cbareiss@olivet.edu, Olivet Nazarene University, Bourbonnais, IL 60914.

Brent Wilson, Northwestern Representative (2021), (503)554-2722, bwilson@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.

Mohamed Lotfy, Rocky Mountain Representative (2019), Information Technology Department, College of Computer & Information Sciences, Regis University, Denver, CO 80221.

Tina Johnson, South Central Representative (2021), (940)397-6201, tina.johnson@mwsu.edu, Dept. of

Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308-2099.

Kevin Treu, Southeastern Representative (2021), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.

Bryan Dixon, Southwestern Representative (2020), (530)898-4864, bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

Serving the CCSC: These members are serving in positions as indicated:

Brian Snider, Associate Membership Secretary, (503)554-2778, bsnider@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.

Will Mitchell, Associate Treasurer, (317)392-3038, willmitchell@acm.org, 1455 S. Greenview Ct, Shelbyville, IN 46176-9248.

John Meinke, Associate Editor, meinkej@acm.org, UMUC Europe Ret, German Post: Werderstr 8, D-68723 Oftersheim, Germany, ph 011-49-6202-5777916.

Shereen Khoja, Comptroller, (503)352-2008, shereen@pacificu.edu, MSC 2615, Pacific University, Forest Grove, OR 97116.

Elizabeth Adams, National Partners Chair, adamses@jmu.edu, James Madison University, 11520 Lockhart Place, Silver Spring, MD 20902.

Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.

Deborah Hwang, Webmaster, (812)488-2193, hwang@evansville.edu, Electrical Engr. & Computer Science, University of Evansville, 1800 Lincoln Ave., Evansville, IN 47722.

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 Partner

Turingscraft
Google for Education
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Bronze Partners

National Center for Women and Information Technology
Teradata
Mercury Learning and Information

Foreword

Welcome to the 2019 issues of our journal for the CCSC spring 2019 conferences: Southwestern (March 22-23), Central Plains (April 5-6), South Central (April 5), Mid-south (April 12-13), and Northeastern (April 12-13).

Please plan to attend one or more conferences, where you can meet and exchange ideas with like-minded computer science educators. Each conference covers a variety of topics that are practical and stimulating. You can find detailed conference programs on the conference websites, which are listed on the CCSC conferencecalendar: `http://www.ccsc.org/regions/calendar.`

From January 2019, this journal will be published electronically on the CCSC website and links to the journal issues will be sent to CCSC members via email. Those of you who would like hard copies of journal issues can order them from Amazon. Simply search for "CCSC Journal" to find available issues. The journal will continue to be available in the ACM Digital Library.

As an author, you may post your papers published by CCSC on any website. Please make sure to use the PDF versions of your papers with CCSC's copyright box. Such PDFs can be downloaded from the ACM Digital Library or extracted from our electronic journal.

Please feel free to email me directly at `blu@sbuniv.edu` if you notice any issue with our publications.

Baochuan Lu
Southwest Baptist University
CCSC Publications Chair

Papers of the 25th Annual CCSC Central Plains Conference

April 5th-6th, 2019
St. Charles Community College
Cottleville, MO

Welcome to the 2019 CCSC Central Plains Conference

2019 ushers in the 25th year of the CCSC Central Plains Region conference and I am proud that St. Charles Community College (SCC) has been selected to host. Every year that I have attended the conference I have taken back to my classroom something new and exciting to help teach and energize my students. Whether it is finding a new tool or technology or partnering with one of my peers, the conference has something for everyone.

This year's conference at SCC we will have some exciting events on the schedule, tours of some of our business partners' facilities, specialty tracks for vendors, students and K-12 teachers to focus in on what interests you most. We will also have a Meet and Greet at the OPO Startup in downtown St. Charles, a technology hub for startup companies in the St. Charles area.

All professional papers, panels, tutorials, and nifty assignments go through a double-blind review process. This year we were able to accept 53% of the paper submissions. We would like to extend our appreciation to the authors who submitted their work for our consideration and to the highly talented group of reviewers that exerted a tremendous amount of time and effort to review all the submissions. Our conference would not be as remarkable without our National Partners, Sponsors, and Vendors for their continued support of our organizations. Thank you all!

Finally, this conference would be nowhere near as successful without the dedication and support from the volunteers and committee members that help support the CCSC efforts throughout the year to bring you this wonderful event. Without their tireless work and dedication, I would not have been able to plan and build a conference this great. This includes the support from past conference chairs, thank you all for letting me be a pest throughout the year of planning.

I hope you have a wonderful time at the CCSC-2019 conference, gain some enlightenment and take back something new to enrich your classroom and research.

Rex McKanry
St. Charles Community College
Conference Chair

2019 CCSC Central Plains Conference Steering Committee

Conference Chair: Rex McKanry ..... St. Charles Community College
Conference Co-Chair: Chetan Jaiswal ..... Truman State University
Conference Publicity: Tom Mertz, Michael P. Rogers ..... Kansas State Polytechnic, Northwest Missouri State University
Keynote Speakers: Scott Bell, Rex McKanry, Chetan Jaiswal ..... Northwest Missouri State University, St. Charles Community College, Truman State University
Pre-Conference Workshop: Judy Mullins, Rex McKanry ..... University of Missouri Kansas City, St. Charles Community College
Papers: Ajay Bandi ..... Northwest Missouri State University
Panels, Tutorials, Workshops: Scott Sigman ..... Drury University
Nifty Assignments: Mahmoud Yousef ..... University of Central Missouri
Lightning Talks: Kendall Bingham ..... University of Missouri Kansas City
K-12 Outreach, Nifty Assignments, Lightning Talks: Scott Bell ... Northwest Missouri State University
Student Paper Session: Scott Sigman ..... Drury University
Student Poster Competition: Joseph Kendall-Morwick ..... Missouri Western University
Student Programming Contest: Charles Riedesel ..... University of Nebraska-Lincoln
Two-Year College Outreach: Rex McKanry . St. Charles Community College
Submission System Admin: David Heise ..... Lincoln University

Regional Board — 2019 CCSC Central Plains Region

Regional Rep Board Chair: Judy Mullins University of Missouri Kansas City
Registrar Membership Chair: Ron McCleary ..... Retired
Current Conference Chair: Rex McKanry ... St. Charles Community College
Next Conference Chair: Chetan Jaiswal ..... Truman State University
Past Conference Chair: Scott Bell ..... Northwest Missouri State University
Secretary: Diana Linville ..... Northwest Missouri State University
Regional Treasurer: Denise Case ..... Northwest Missouri State University
Regional Editor: Bin "Crystal" Peng ..... Park University
Webmaster: Michael P. Rogers ..... Northwest Missouri State University

Reviewers — 2019 CCSC Central Plains Conference

Imad Al Saeed ..... Saint Xavier University, Chicago, IL
Rad Alrifai ..... Northeastern State University, Tahlequah, OK
Beth Arrowsmith ..... University of Missouri St. Louis, Saint Louis, MO
Ajay Bandi ..... Northwest Missouri State University, Maryville, MO
Scott Bell ..... Northwest Missouri State University, Maryville, MO
John Buerck ..... Saint Louis University, Saint Louis, MO
Denise Case ..... Northwest Missouri State University, Maryville, MO
Chia-Chu Chiang .... University of Arkansas at Little Rock, Little Rock, AR
Tim DeClue ..... Southwest Baptist University, Bolivar, MO
George Dimitoglou ..... Hood College, Frederick, MD
Aziz Fellah ..... Northwest Missouri State University, Maryville, MO
Ernest Ferguson ..... Northwest Missouri State University, Maryville, MO
David Furcy ..... University of Wisconsin Oshkosh, Oshkosh, WI
David Heise ..... Lincoln University, Jefferson City, MO
Suvineetha Herath ..... Carl Sandburg College, Galesburg, IL
Charitha Hettiarachchi . Northwest Missouri State University, Maryville, MO
Wen Hsin ..... Park University Parkville, MO
Chetan Jaiswal ..... Truman State University, Kirksville, MO
Yipkei Kwok ..... Missouri Western State University, St. Joseph, MO
Baochuan Lu ..... Southwest Baptist University, Bolivar, MO
Rick Massengale ..... Arkansas Tech University, Russellville, AR
Ron McCleary ..... Retired
Bruce Mechtly ..... Washburn University, Topeka, KS
Jose Metrolho ..... Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
Michael Oudshoorn ..... High Point University, High Point, NC
Michael Rogers ..... Northwest Missouri State University, Maryville, MO
Jamil Saquer ..... Missouri State University, Springfield, MO
Matthew Schieber ..... Northwest Missouri State University, Maryville, MO
William Siever ..... Washington University in St. Louis, Saint Louis, MO
Jeffery Solheim ..... Fort Hays State University, Hays, KS
Carol Spradling ..... Northwest Missouri State University, Maryville, MO
Timothy Urness ..... Drake University, Des Moines, IA
Nancy Van Cleave ..... Eastern Illinois University, Charleston, IL
Ken Vollmar ..... Missouri State University, Springfield, MO
Henry Walker ..... Grinnell College, Grinnell, IA
Baoqiang Yan ..... Missouri Western State University, St. Joseph, MO
Mahmoud Yousef ..... University of Central Missouri, Warrensburg, MO

STEM and Future Workforce[1]

Friday Opening Keynote

Randy Schilling
Founder of BoardPaq and OPO Startups

Schilling is the President and Chief Executive Officer of BoardPaq, an affordable, easy-to-use, secure Board of Directors iPad and web (PC/Mac) application for planning, running and managing paperless Board meetings. The management team at BoardPaq has more than 65 years of experience in the IT industry. The Company's offices are located in Historic Downtown Saint Charles, Missouri.

Boardpaq is dedicated to providing services to improve overall board functionality. BoardPaq is more than a meeting facilitator; it seeks to provide an effective communication pathway for more engaged and productive members. Boardpaq offers a paperless environment that yields more efficient communication. Boardpaq is the ultimate collaboration tool that empowers boards to be great in and out of the boardroom.

Schilling is the former Chief Executive Officer and Founder of Quilogy, a Microsoft nationally managed system integrator, which is located in historic St. Charles, Missouri. Schilling founded Quilogy in 1992 under the name Solutech, Inc. with a personal investment of $5,000. Under Schilling's leadership, the company grew to over 500 employees and 16 offices.

Schilling is a long-time resident of St. Charles, Missouri, his hometown. He is active in a wide range of business and civic organizations, including: Director, CASS Information Systems, (NASDAQ: CASS), the nation's leading provider of transportation, utility and telecom invoice payment and information services, Chairperson of Education Committee of St. Charles County Partners for Progress, a group of leading business and community leaders that manage and encourage commerce and growth in St. Charles County, and Advisory Council Member, Arch Grants, a non profit organization that accelerates economic development by providing $50,000 equity-free grants and pro bono support services to entrepreneurs who locate their early-stage business in St. Louis, Missouri.

Debugging Diversity: Creating Opportunities through Computer Science Education[1]

Banquet Keynote

Sherea Dunlap
Executive Director at Create a Loop

Sherea Dunlap began teaching in Baltimore City after earning a Master's Degree from University of Maryland, College Park. On moving to St. Louis, Sherea began a six-year career teaching upper elementary and middle school at St. Louis independent schools: Community School and City Academy.

At Create a Loop, Sherea's focus is on bringing AP Computer Science and Washington University of St. Louis college-level course material to Create a LOOP's elementary, middle and high school curriculum.

Microsoft, LinkedIn, Online Professionalism[1]

Saturday Opening Keynote

Jo Otey
Customer Success Manager, Microsoft

Otey is a technology guy who puts people before machines, specializing in solving problems of digital transformation within healthcare, nutrition, startups, and non-profits. Working at Microsoft, he has access to numerous resources and professional experts to solve complex problems. He believes when you have a growth mindset anything is possible.

Revising an Accredited Computer Science Program at a Public Regional University to Meet New ABET Guidelines*

Xiaodong Yue, Belinda Copus, Hyungbae Park
Mahmoud Yousef and Songlin Tian
School of Computer Science and Mathematics
University of Central Missouri
Warrensburg, MO 64093
{yue,copus,park,yousef,tian}@ucmo.edu

Abstract

In response to the most recent ABET CAC accreditation criteria change, the computer science program at the University of Central Missouri revised its curriculum and assessment process in preparation for a re-accreditation visit in Fall 2019. In this paper, a case study is presented based on the lessons learned and experiences gained from our revision process. The findings and recommendations summarized in this paper could be adopted in a similar setting by other institutions, especially those with re-accreditation visits in Fall 2019 or beyond.

1 Introduction

The new ABET Computing Accreditation Commission (CAC) accreditation criteria[1] were approved in October 2017. The new criteria are optional for reviews during the 2018-2019 accreditation cycle, but are mandatory for the 2019-2020 accreditation visit. In preparation for its re-accreditation visit in Fall 2019, the computer science program at the University of Central Missouri

revised its curriculum and continuous improvement process in the 2017-2018 academic year to comply with the new criteria. We hope that other computer science programs with a re-accreditation visit in Fall 2019 or beyond will benefit from the experiences and findings discussed in this paper.

2 Background

The BS in Computer Science (CS) at the University of Central Missouri was accredited by ABET CAC under the computer science program criteria. The CS program is a 120-credit hour program, composed of 36 hours of required computer science courses, 6 hours of computer science electives, 30 hours of mathematics and science, 37-44 hours of general education, and 4-11 hours of free electives.

3 Meeting The New Criteria

There are significant changes in the new ABET CAC accreditation criteria, especially in Criterion 3, Student Outcomes; Criterion 5, Curriculum; and in the computer science program criteria. In the next section we will go through each of the criteria one by one to provide a synopsis on our understanding of the key issues to be addressed and areas needing special attention.

3.1 Criterion 3: Student Outcomes

The current CAC criterion 3 has 9 student outcomes (a-i). Under the current criterion, a program is allowed to define its own outcomes as long as those outcomes can be mapped to CAC outcomes a-i. The new CAC criterion 3 reduces the number of student outcomes from 9 to 5. In addition, all 5 new student outcomes are required for any computing program. A program may define additional outcomes. Since the new ABET CAC student outcomes are fairly comprehensive, the faculty determined that no additional outcomes are needed for our CS program. Since all the student outcomes are required, ABET no longer requires periodic review of student outcomes. As a result, the new CAC self-study questionnaire[2] on Criterion 3 has been significantly simplified. In particular, the program is no longer required to describe 1) how the student outcomes prepare graduates to attain the program educational objectives (PEOs); 2) the process used for reviewing and revising student outcomes; and 3) how programs enable students to attain, by the time of graduation, characteristics (a) through (i) as listed in Criterion 3 as well as any applicable characteristics defined within the program criteria. Instead, programs are now

required to describe how the student outcomes are documented and publicly stated. This new requirement is easily fulfilled by posting to the program website. In summary, our finding is that Criterion 3 is significantly simplified in the new criteria.

3.2 Criterion 5: Curriculum

The minimum credit hours requirement in computing topics remains 30 in the new CAC criteria Criterion 5. The most significant change is that, besides at least 30 semester credit hours (or equivalent) of up-to-date coverage of fundamental and advanced computing topics that provide both breadth and depth, programs must include computing topics from the following 3 areas:

1. Techniques, skills, and tools necessary for computing practice;
2. Principles and practices for secure computing;
3. Local and global impacts of computing solutions on individuals, organizations, and society.

Topic 1 is related to the current CAC student outcome i. Since techniques, skills, and tools necessary for computing practice are covered in various computer science courses in our current program, this new requirement can be met without any program revision.

There are several secure computing-related elective courses in our current CS program. By carefully reviewing course content and associated prerequisites, the faculty determined that an existing elective course in information assurance should become a required course, as it provides balanced coverage of various secure computing concepts. This modification required minimal adjustment to the curriculum and did not increase required credit hours.

Topic 3 is associated with current CAC student outcome g. Those topics are covered in a required ethics course in the current CS program. As a result, this new requirement can also be met without any program revision.

The new CAC self-study questionnaire on Criterion 5 has no significant changes with respect to the current questionnaire except for some minor editorial changes. In summary, our finding is that, although there are several changes in the new CAC Criterion 5, it should be fairly easy for programs to adjust in terms of curriculum revisions.

3.3 Computer Science Program Criteria

The computer science program criteria, in our opinion, have the most significant changes. The current CAC criteria adds two additional student outcomes, j and k, for computer science programs, while the new criteria reduces the number of program specific student outcomes to a new single outcome. This new

student outcome is also required for every computer science program. Overall, the total number of student outcomes has been reduced from 11 in the current CAC criteria to 6 in the new CAC criteria for computer science programs. Our conclusion is that such change will significantly reduce the amount of assessment and evaluation effort required by program faculty, and will thus result in a more sustainable and effective program improvement process.

3.3.1 Computer Science Requirements

The new CAC computer science program criteria requires at least 40 semester credit hours (or equivalent) that must include:

1. Substantial coverage of algorithms and complexity, computer science theory, concepts of programming languages, and software development;
2. Substantial coverage of at least one general-purpose programming language;
3. Exposure to computer architecture and organization, information management, networking and communication, operating systems, and parallel and distributed computing;
4. The study of computing-based systems at varying levels of abstraction;
5. A major project that requires integration and application of knowledge and skills acquired in earlier course work.

The minimum credit hours requirement in computer science course work remains 40 in the new CAC criteria. The requirements change from 4 items in the current CAC criteria to 5 items in the new CAC criteria.

Requirement 1 in the new criteria corresponds to Requirement 1 in the current criteria. There are several changes in this requirement. First of all, instead of "coverage" in the current criteria, the new criteria emphasizes "substantial coverage." Second, "fundamentals of algorithms" and "software design" are revised to "algorithms and complexity" and "software development," respectively. Third, "data structures" and "computer organization and architecture" are replaced by "computer science theory."

Although data structures is no longer part of this requirement, the faculty decided to retain the Data Structures course as a required course in the CS program since it covers some fundamental algorithms, computing theory, and extensive Java programming experience. In response to the deletion of the computer organization and architecture topics, the faculty decided to remove a previously required course, Introduction to Computer Organization, and integrate parts of the content with another course. By reviewing the current CS program, the faculty concluded that our current program already provides substantial coverage of algorithms and complexity and computer science theory. Therefore, no adjustments were required for those topics. Our interpretation of the change from Software Design to Software Development is that Software De-

sign is a process of problem-solving and planning for a software solution while Software Development focuses more on implementation of a software solution and its whole life cycle. The faculty concluded that the current software-related courses in the CS program could be retained to meet this requirement.

Requirement 2 in the new criteria corresponds to Requirement 3 in the current criteria. There are two changes in this requirement: First, the words "proficiency in" are replaced by "substantial coverage of." Second, "higher-level language" is replaced by "general-purpose programming language." Our interpretation is that this requirement has been moderated in the new criteria since the words "substantial coverage of" are weaker than "proficiency in." Programs can always provide substantial coverage of a programming language, but cannot always guarantee students' proficiency in that language. In addition, the primary instructional language in our CS program is Java—a general-purpose programming language. There are 3 required courses and several elective courses in the current CS program that provide significant coverage of Java. As a result, our conclusion is that this requirement is met by the current CS program. It is worth noting that since the new criteria emphasize "substantial coverage," our approach is to make sure there are at least several required courses, coupled with a few electives, covering each topic.

Requirement 3 in the new criteria corresponds to Requirement 2 in the current criteria. Instead of the generic wording in the current criteria, "a variety of programming languages and systems," this requirement now has a very specific list of curricular topics: computer architecture and organization, information management, networking and communication, operating systems, and parallel and distributed computing.

The current CS program has two required courses, Database and Operating Systems, in which information management and operating systems topics are covered. We combined our Computer Architecture course with our Computer Organization course and retitled it to, "Computer Organization and Architecture." The course content was redesigned so that the revised course provides a balanced coverage of both organization and architecture. Since the current CS program does not have a required course in networking and communication, the faculty decided to change an elective course in Computer Networking course to a required course. With the old Computer Organization course removed, no extra credit hours needed to be added to the program. We also determined that the parallel and distributed computing concepts are taught in three required courses that already exist. It is also worth noting that programs need only provide "exposure" to topics listed in this requirement and most of these topics are already included in existing required courses.

Requirements 4 and 5 are both new. By reviewing the CS program, the faculty concluded that Requirement 4 is met by the concepts covered in several

required courses, as is Requirement 5.

3.3.2 Mathematics and Science Requirements

One significant change in the Mathematics and Science requirement is that the minimum 30 credit hours requirement is lifted in the new CAC criteria. The minimum 15 credit hours in mathematics remains intact. Our current CS program requires a minimum 16 credit hours in mathematics and no program adjustment is needed.

To address the reduced credit hours in Science, the faculty proposed to reduce the science credit hours from 14 to eight so that students will only need to take two science courses with labs. Three of the six reduced hours were applied to the newly required Information Assurance course, mentioned in section 3.2. The remaining three applied as elective hours, which resolved a common student complaint that there are not enough elective hours in the current CS program.

The revised CS program remains a 120 credit hours program, comprised of 36 hours in required computer science courses, 9 hours in computer science electives, 24 hours in mathematics and science, 37-44 hours in general education, and 7-14 hours in free electives. The revised program is more flexible when compared with the current program, with more hours in computer science electives and in free electives. The faculty believe these changes may attract and retain more students in the CS program.

4 Conclusions

The changes to the ABET CAC criteria are significant, and all computer science programs must revise their curricula and assessment plans before their next visits. Based on our experience, this is something that can be accomplished even in a short time frame given sufficient planning and faculty involvement while providing a context for training younger faculty in assessment and the continuous improvement process.

References

[1] ABET. Criteria for accrediting computing programs, 2018-2019, version 2. `https://www.abet.org/wp-content/uploads/2018/02/C001-18-19-CAC-Criteria-Version-2.0-updated-02-12-18.pdf`.

[2] ABET. Self-study questionnaire: Template for the computing self-study, 2018-2019, version 2. `http://www.abet.org/wp-content/uploads/2018/03/C002B-CAC-Self-Study-Questionnaire-2018-19-Version-2.0-FINAL.docx/`.

The Visible File System – An Application for Teaching File System Internals*

Bruce Mechtly, Fritz Helbert, Dylan Cox, and Zachary Hastings
Washburn University
Topeka, KS 66621
bruce.mechtly@washburn.edu

Abstract

In digital forensics one sometimes needs to look at a file system without filtering. One may be interested in seeing items that have been deleted, or see how deletion is done in a particular file system. One may also be interested in looking at slack space, or the space between partitions. The Visible File System (VisFS) application was developed to allow unfiltered viewing of file system structures as well as searching for patterns in those structures and file blocks. The main view shows the raw disk content in hexadecimal as well as a collapsible tree showing the meaning of the data. Currently, VisFS supports VFAT, NTFS, Ext, HFS, ISO and UDF file systems. VisFS also automatically indexes all blocks that are allocated in the directory tree so that blocks identified in a search can be easily found in the directory tree. VisFS runs on Windows, Mac, and Linux.

1 Introduction

This work began while preparing an introductory digital forensics class. It was difficult to visualize how deleted items appear in a file system directory. At first hex dumps of the directory, blocks were shown before and after deletion with hand-drawn markup showing the file name, timestamps, links, and other

metadata. It occurred to the instructor that a Java application could automate the process and provide links from directory blocks to other directory and file blocks.

The VisFS application now allows visualization of the Master Boot Record (MBR), GPT (GUID Partition Table) and the structures and files of six types of file systems. It can also make dd (raw) images of devices and partitions.

The main window consists of two sections. On the left is a JTree object that organizes the data into a collapsible tree. By collapsing tree nodes one can select more or less detail. On the right is a hex dump of the raw data on the device (usually one sector at a time). By using buttons one can step forward or backward and view a hex dump of surrounding sectors.

In the tree view items marked with a "*" at the end can be double-clicked to follow links. In this way one can start in the root directory and follow any path through the file system.

The "Search" tab allows searching by ASCII/UTF-8, UTF-16 (big and little-endian), hex and regular expressions. When the target is found the sector appears in a list with a link to the directory tree if the sector is in an allocated block.

2 Master Boot Records and Partition Tables

Figure 1 shows the contents of a master boot record (MBR) with a DOS-style partition table. Note that the second partition is selected in the Tree View and highlighted in the hex dump. If any extended partitions are present there are links to follow the EBR (extended boot record) chain.

VisFS can also interpret GUID partition tables (GPT). In this case, the MBR is called a "Protective MBR" and is retained for legacy systems.

3 The VFAT (FAT-32) File System

The first sector of a VFAT file system is called the volume boot record (VBR). It contains information about the partition such as the number of sectors per cluster, the size of the FATs, etc. From this information, you can calculate where the data clusters start. The root directory is always at data cluster 2. Figure 2 shows the root directory block for a VFAT partition.

Note that one directory entry is expanded. The last two lines are the DOS directory entry containing the DOS 8.3 file name, timestamps, and starting data cluster. The first two lines are the VFAT long file name "VisibleFS.jar". The long file names work backward from the directory entry. The first nibble

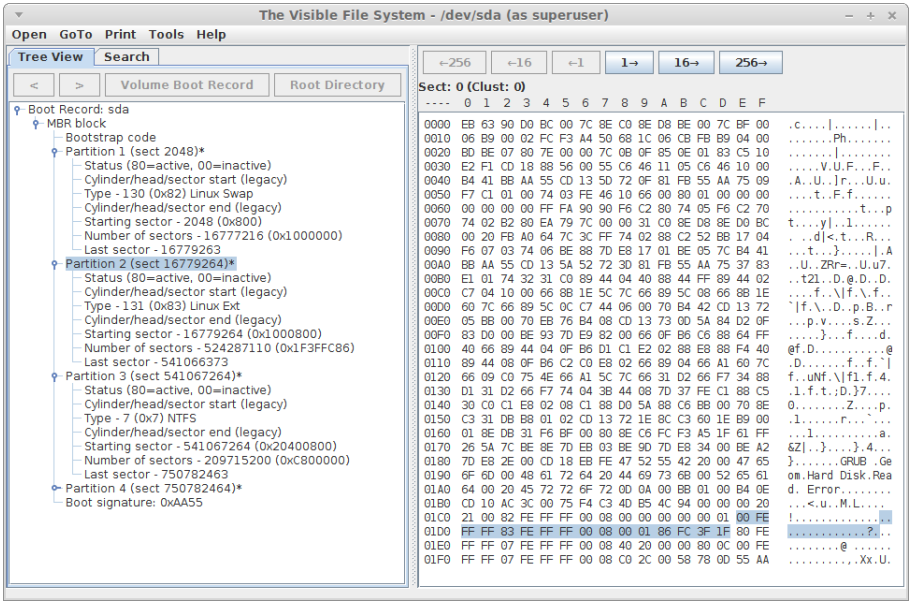


Figure 1: A Classic MBR with DOS-style Partition Table

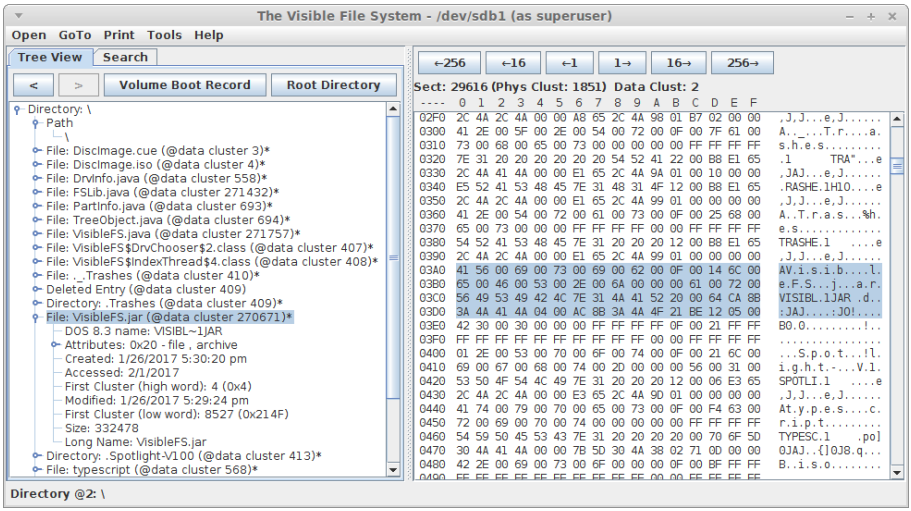


Figure 2: The Root Directory for a VFAT Partition

of the first byte is a 0 or a 4 (4 indicating the last one). The second nibble is a sequence number.

Deleted entries are marked with an E5 in their first byte. This partially obscures the DOS 8.3 file name, but timestamps and starting data cluster remain intact. This may have some forensic value.

Double-clicking on a directory entry for a file will take you to a listing of the file blocks. The blocks are found by following links in the FAT. If the blocks are contiguous they are listed as a "data run".

4 The NTFS File System

The first sector of an NTFS file system is called the BIOS parameter block (BPB). It gives information about the partition such as the size of MFT (Master File Table) and INDX (Directory B-Tree Nodes) blocks. The root directory is always pointed to by MFT #5.

Figure 3 shows the root directory of an NTFS file system. Note that an MFT is organized into attributes. Only directories have $INDEX_ROOT and $INDEX_ALLOCATION attributes. The $INDEX_ROOT attribute is the root node of a B-Tree for that directory. Small directories may be complete in the $INDEX_ROOT attribute of the MFT. Larger directories will have links to INDX blocks which are child nodes in the B-Tree.

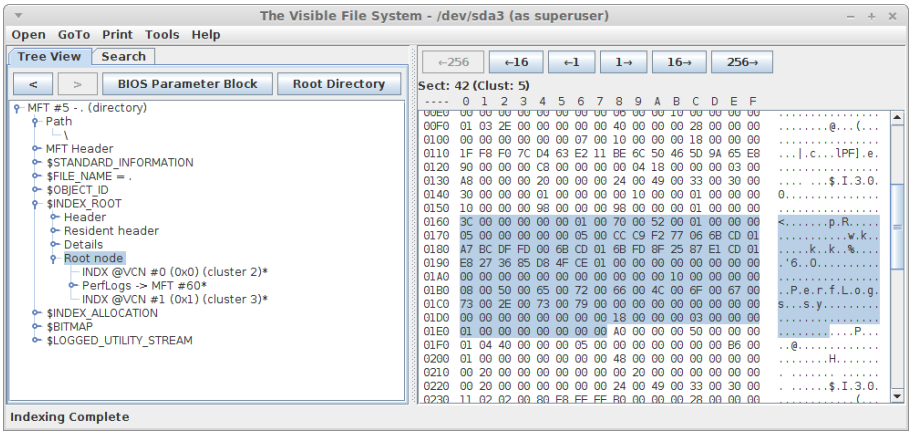


Figure 3: The Root Directory of an NTFS File System

Figure 4 shows an INDX block. INDX blocks are simply lists of directory entries. Each entry points to an MFT block. In this case, the INDX block is a

B-Tree leaf so it doesn't contain any INDX links. In a larger directory, there would be several levels of non-leaf nodes.

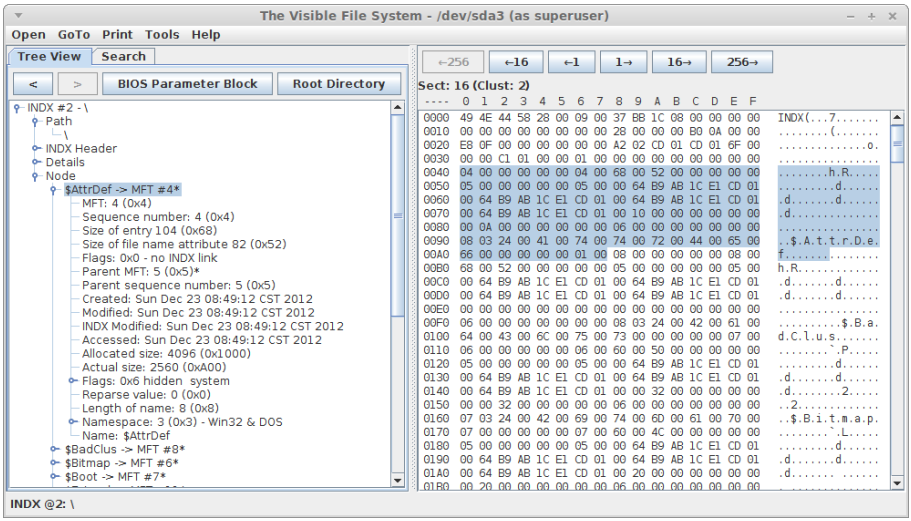


Figure 4: An NTFS INDX Block

Note that NTFS is rich with metadata. Each directory entry has four timestamps. Each MFT has at least eight timestamps (four in the $STANDARD_INFORMATION attribute and four in the $FILE_NAME attribute. Often an MFT will have more than one $FILE_NAME attribute.

Because of how B-Tree nodes split there are sometimes old directory entries that are no longer used (the second half of the directory entries are rewritten into a new node). The old data is not cleared. This means that there may be old file names and timestamps left over from a previous state of the file system.

5 Searching

Figure 5 shows the Search tab. The entire partition is searched for a target string. The matched sectors are listed at the bottom. Single-clicking on one of them will show the target in the relevant sector. Double-clicking will take you to the relevant data structure in the directory tree (if the sector is indexed).

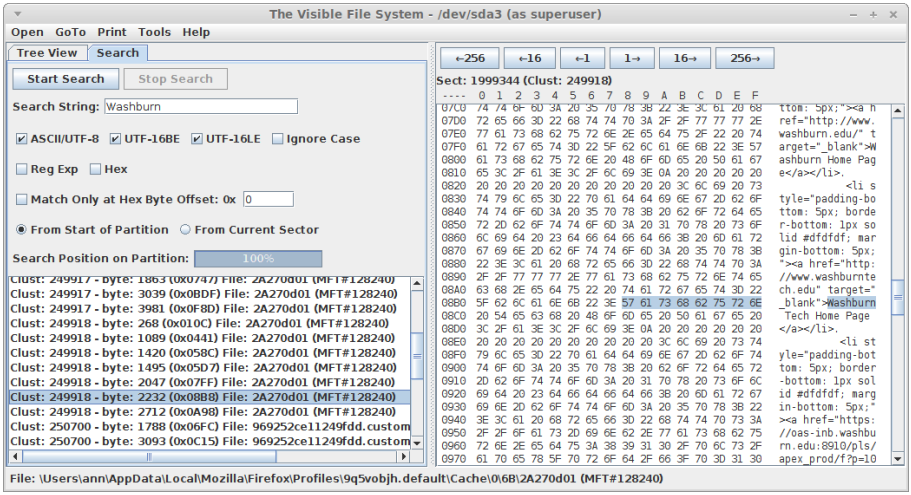


Figure 5: The Search Tab

6 Other File Systems

There is insufficient space in this paper to show examples of other file systems. A brief description must suffice.

Ext file systems are based on I-nodes. Directories are stored in H-Trees (similar to a B-Tree but based on hashes of file names). I-nodes are distributed in groups over the device (to minimize R/W head motion). The root directory is always at I-node #2. Ext2 and Ext3 have the classic UNIX single, double and triple indirect pointers. Ext4 uses "extents" instead of a list of pointers. Each extent is a pair of integers giving the starting block and run length.

The HFS (hierarchical file system) is the file system used on MACs. The directory tree is stored in a "Catalog File". It is a B+-tree with all folder and files listed serially at the lowest level. The B+-tree nodes at the lowest level are also connected in a linked list. Hierarchical relationships between folders are provided through "Catalog ID Numbers" or CNIDs. Each catalog node has room for eight extents. If more extents are required these are contained in the "Extent Overflow File".

ISO-9660 is a file system for data CDs and DVDs. It is a simple file system where files are assumed to be contiguous (originally designed for read-only media). Later on, features were added to allow the file system to be modified (CDRW, etc), but in reality, the modifications are written on a different place on the disk and a new directory tree is written that supersedes the old one. These are called "sessions". A forensics tool should allow the investigator to

see older sessions so all the data on the disk can be examined. Extensions for non-DOS 8.3 file names were added later. VisFS shows all sessions on the disk.

UDF (Universal Disk Format) is a much more advanced system for data CDs, DVDs and Blu-Rays. It supports multiple sessions in a novel way. With each session, a Virtual Allocation Table (VAT) is written which remaps existing blocks to new blocks when items are modified. Only the last VAT is current, but in a forensics context, an investigator may want to use the older VATs. VisFS can detect multiple sessions and allow the investigator to select which VAT to use.

7 Design Details

No file system libraries are needed to run VisFS. This decision was made so that it could run on Windows, Mac, and Linux without dependencies. Devices are opened using Java's RandomAccessFile class. Windows doesn't report non-Windows partitions, so the entire device is opened as a RandomAccessFile and the VisFS application keeps track of where the partitions start internally. Unfortunately, raw device access requires admin or root access. Image files do not require admin access and are a good choice for students in a lab where they don't have such privileges.

The JTree object is a collection of TreeNode objects. We used the DefaultMutableTreeNode class where we passed a TreeObject object to its constructor. The TreeObject class is one that we created to store information that connects the tree to the hex dump. It contains the text displayed in the tree node as well as a command string (what to do if the user double-clicks on the node), integers to determine where highlighting in the hex dump should begin and end, and a boolean to determine if the node should be collapsed or expanded.

Many file system specification references were consulted. Brian Carrier's book[2] is one of the best sources available. Most of the other resources were web-based (VFAT[4], NTFS[7], Ext[5], HFS[3], ISO-9660[6], UDF[1]. Dozens of other web-based resources were used but there is not enough space to list them here.

Note: The Visible File System application can be downloaded from the following URL: `http://cislinux2.washburn.edu/ForensicsTools/VisibleFS.jar`

References

[1] Optical Storage Technology Association. Specifications: Universal disk format (UDF). `http://www.osta.org/specs/`.

[2] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley, Reading, Massachusetts, 2005.

[3] Matt Deatherage, Justin Seal, Nathaniel Irons, Jerry Kindall, John C. Welch, and John Gruber. The HFS primer. `http://macjournals.com/~mwj/mwj_samples/MWJ_20030525.pdf`. Retrieved September 27, 2017.

[4] Author unknown. Design of the FAT file system. `https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system`. Retrieved September 27, 2017.

[5] Author unknown. Ext4 Disk Layout. `https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout`. Retrieved September 27, 2017.

[6] Author unknown. ISO 9660. `http://macjournals.com/~mwj/mwj_samples/MWJ_20030525.pdf`. Retrieved September 27, 2017.

[7] Author unknown. NTFS. `https://en.wikipedia.org/wiki/NTFS`. Retrieved September 27, 2017.

Initial Evaluation of Accessibility and Design Awareness with 3-D Immersive Environments *

Nicholas Rosasco[1], Alex Kaariainen[2], Jeffrey Will[2]
[1]Department of Computing and Information Sciences
[2]Department of Electrical and Computer Engineering
Valparaiso University
Valparasio, IN 46383
{nicholas.rosasco,alex.kaariainen,jeff.will}@valpo.edu

Abstract

This paper describes an effort to build and evaluate the effectiveness of an immersive 3-D visualization system to help increase the awareness that students have when designing software that has a high level of accessibility for the differently abled. The demonstration utilizes an immersive virtual reality (VR) environment in which we simulated two types of colorblindness in a generally familiar environment. We report on the initial trial of this tool and the results of student surveys designed to assess impact on student perception and understanding and demonstrate that the use of virtual environments can give students greater empathy for individuals with visual impairments.

1 Introduction

In the United States, the Americans with Disabilities Act (ADA) [5] and subsequent legislation defines a statutory requirement for accessible software in the private sector. Further regulations, including Section 508 [6], bolster these expectations for software and systems provided by the national government. Similar legal and regulatory structures exist in multiple jurisdictions, as shown

through the European Accessibility Act, various Japanese regulations [17], and laws found everywhere from Europe to the People's Republic of China, among many others. Globally, this standard is so general an expectation that an index for policies covering web-based systems alone is maintained by the W3C [18].

This legal expectation is paralleled by curricular, ethics, and policy guidance provided by and for the computing disciplines. The ACM/IEEE CS2013 [3] curricular framework provided by the combined professional and academic computing community includes multiple mentions of accessibility concerns. It is a foundational, core/tier one topic in "Human Computer Interaction". Knowledge in this topic is also noted as a tier two area under the Social Context heading. Specifically, CS2013's expectations on developer assumptions and accessibility are found in "SP/Social Context": "Identify developer assumptions and values embedded in hardware and software design, especially as they pertain to usability for diverse populations including under-represented populations and the disabled." Related concepts can also be found in the sections for "HCI/Foundations" and "HCI/Designing Interactions":

- Define a user-centered design process that explicitly takes account of the fact that the user is not like the developer or their acquaintances.
- Discuss why human-centered software development is important.
- For an identified user group, undertake and document an analysis of their needs.

This doubled imperative – legal and professional - creates a need to effectively impart the significance of these requirements and expectations upon the students. Unfortunately, it is understood that not all topics listed may be covered, as detailed in the CS2013 "Principles" discussion as well as the Chapter 4 introduction.

In order to make the most of the time available, it may be possible to combine these areas with additional learning objectives. Specifically, these can be linked to the overall importance of design as a way to anticipate and solve problems. By doing this, the need for accessibility awareness can be articulated as functional and non-functional requirements. In turn, these requirements can be discussed as needs that must be met for any software or system a student may build as a working professional.

By recasting this issue into the context of an expectation, it is possible to concurrently touch on multiple CS2013 sections that refer to the design process, particularly with reference to user interfaces, including:

- Explain how user-centered design complements other software process models,
- Choose appropriate methods to support the development of a specific UI, and

- Use a variety of techniques to evaluate a given UI.

This also provides an opportunity to make this expectation a habitual, as opposed to an occasional, consideration – and to link it to good design generally. Similarly, the importance of effective design is emphasized both in the curricular guidance - and in the commonly accepted guidance for software engineering practice. While the fines or repercussions of failing to provide the differently abled will vary by jurisdiction, the use of software engineering solutions at the design stage is general. Similarly, the convention is that a mistake made becomes an order of magnitude costlier with each subsequent round (design, implementation, test, etcetera) in general [2][4].

2 Background, Context, and Motivation

Various studies, including significant recent work from ongoing projects at Stanford [9][7][15], provide evidence that virtual reality environments can be used to foster empathy and understanding. Additionally, previous scholarship in teaching computer science and software engineering provides numerous examples for the teaching of accessibility issues, and fostering awareness [19]. Much of this instruction is placed in the context of a human-computer interface or other specialized elective course. The work of Shinohara et al illustrates that this is a general pattern for this topic across the United States [10]. This could imply that, for institutions without an accessibility specialist, or with the ability to offer limited electives, this topic is at risk of neglect. The same study indicates that curricular integration was still a central issue. To help meet this need to increase the awareness of accessibility issues, it was decided to add a CAVE system to the collection of examples and tools used for teaching this topic.

To meet that need, a project at Valparaiso University simulated two types of colorblindness in a generally familiar environment. We then used a student survey to assess the impact on the student's perception and understanding and demonstrate that the use of virtual environments can give students greater empathy for individuals with visual impairments. The project sought to incorporate these objectives into a session that could be incorporated into required courses. This was considered especially important as the size of Valparaiso's program dictates a smaller breadth of electives than what would be available at a larger institution or from a larger department. It was felt that the best possible presentation would put the inclusion of accessibility into a larger narrative. By linking accessibility to design, and by making designing for accessibility a preemptive solution that could prevent an expensive retrofit or required corrective re-release, a narrative showing the power of thinking ahead can be built.

The project described and explained in this paper is intended to see if a memorable and suitable moment in a classroom can be built to accommodate those background factors. The expectation is that the use of an atypical technology can create an understanding and impression that can impact a student's thinking about design and accessibility. That experience was be built around colorblindness.

Colorblindness was chosen as a consideration to address because it is an issue found in a significant part of the general public [8], occurs in varying degrees, and is not externally apparent that someone has the physical issue [11]. For example, the National Eye Institute [8] notes that, "As many as 8 percent of men and 0.5 percent of women with Northern European ancestry have the common form of red-green color blindness." Also, in the United States during 2005, it is estimated that 1 percent or about 1.46 million men suffer from deuteranopia [14]. For the purposes of classroom use, this makes it likely that every student may either have or have a close connection to someone with this condition. Two forms of the condition, protanopia and deuteranopia, were chosen for demonstration and simulation. They are commonly referred to as red-green color blindness [8][1].

3 Methodology

Valparaiso University's Electrical and Computer Engineering Department owns and operates a commercially available CAVE Virtual Reality System ("VisCube") environment developed by Visbox, Inc [16] as shown in Figure 1. This environment can immerse six to eight people simultaneously and is thus effective as an educational tool. The four displays are merged to create a seamless 3 D environment that makes use of the entire space, including the floor as well as the three front facing walls. This creates a similar experience to that of the more popular head mounted displays (HMDs).

For this project, protanopia and deuteranopia were simulated via a post-processing effect [13] using a 3D look up table (LUT) in the Unity Engine [12]. Three unique post-processing profiles were used, two that illustrate colorblindness, the third for the 'unaltered vision mode' because various visual improvements were still necessary such as anti-aliasing and ambient occlusion. Switching between these three configurations, which can be triggered with a remote control, allow at run-time swapping between the post-processing profiles. Both instructor and students share the same VR experience. This quick-change capability, combined with a brief discussion meant that 24 students - a standard section size at Valparaiso University - were able to have an immersive and educational experience in the space of a 50 minute 'Carnegie hour' class period.

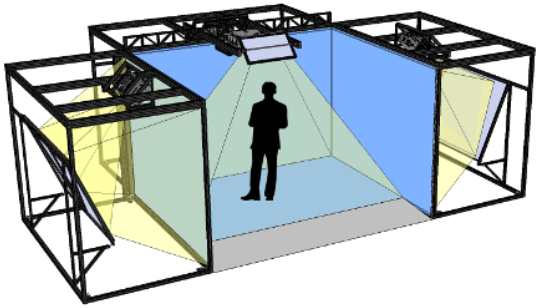


Figure 1: VisCube 3D Cave Environment

To test the impact of this exercise, Institutional Review Board approval was sought to use a pre- and post- exercise survey on current students as test subjects. For this initial study, students in the 'Seminar in Professional Practices' course were the participants. This course covers topics such as ethics, computing's impact on society, and other questions related to professional and academic practices. It is traditionally taken by upperclassmen, many of whom have had REU or internship experiences and most of whom have spent significant time developing software. Some students had participated in some initial rollout exercises of this same configuration, so had seen an earlier pilot edition of the overall sequence and presentation. For this initial trial, 14 students, approximately three quarters of a graduating class for Valparaiso University's Computer Science program, participated. For the first trial of the system and lesson, a set of pre-survey questions was given to the participants. As the development of this suite involved some exposure to the tools prior to the first structured, intentional test, a question about for prior experience was included. After that check and a request for explanation, the following questions were posed:

- Have you experienced simulated or actual colorblindness previously?
  (The remaining questions include Likert-scaled responses.)
- Your awareness of the need to consider colorblindness when designing software and systems.*
  1/Not at all aware, 2/Slightly aware, 3/Somewhat aware, 4/Moderately aware, 5/Extremely aware
- Your awareness of the need to design software and systems for the differently abled, in general (not just for colorblindness).*
  1/Not at all aware, 2/Slightly aware, 3/Somewhat aware, 4/Moderately aware, 5/Extremely aware
- Do you regularly consider the needs of the differently abled when coding or

building applications?
1/Never, 2/Rarely, 3/Sometimes, 4/Often, 5/Always
(Questions marked with an asterisk at the end were posed before and after the exercise.)

Once a group of students had completed the pre-survey, we asked them to accompany the instructor and enter the VisCube to immediately begin the demonstration. The demonstration begins in a rendering of a gymnasium. Various attributes were intended to create emphasis of color. For example, a basketball and a soccer ball were placed on the gymnasium floor as their colors are something that is familiar to most students. The first switch from 'unaltered' mode to each of the colorblindness modes was done in that space and used to start a conversation about perception. During the demonstration, the instructor leads a "walking and talking" discussion. The instructor then leads the group into several spaces configured for various STEM disciplines which are used to illustrate some challenges colorblind students may face in the classroom. For example, while in the chemistry classroom the instructor makes a point of the periodic table poster which relies heavily on color to convey various information. Then, several discussion questions – with the system set for the less and then the more challenging degree of colorblindness – are posed to move the discussion to a close. The final set of questions reframes the issues the students had have reading a specific item, the periodic table, as a factor in a high stress/high stakes situation like a test. The discussion is concluded as the group returns to the starting point, leaving the demonstration ready for the next group.

Students were then asked several questions as a post-survey and again asked to respond on a Likert scale:

- Your awareness of the need to consider colorblindness when designing software and systems?
  1/Not at all aware, 2/Slightly aware, 3/Somewhat aware, 4/Moderately aware, 5/Extremely aware
- Your awareness of the need to design software and systems for the differently abled, in general (not just for colorblindness).*
  1/Not at all aware, 2/Slightly aware, 3/Somewhat aware,4/Moderately aware, 5/Extremely aware
- Will you regularly consider the needs of the differently abled when coding or building applications?
  1/Never, 2/Rarely, 3/Sometimes, 4/Often, 5/Always
- Did the immersive experience in the VisCube visualization system increase your knowledge of the need to build software with this issue in mind?
  1/No impact on me, 2/Minor impact on me, 3/Neutral, 4/Moderate impact on me, 5/Major impact on me
- Do you think you'll remember this experience longer term?

1/Never-No, 2/Probably not, 3/Occasionally-Sometimes, 4/Lots-Usually, 5/Always-almost always

They are also given two additional comments that allow for more general comments and feedback:

- Any other comments or observations, generally?
- Any thoughts/observations on the demo, specifically?

4 Results

Our initial trial included 14 student participants. While this is a somewhat small number, it approximates a graduating class from the Computer Science programs at Valparaiso University. Our sample population included some students with prior direct or indirect exposure to this demonstration for various reasons. To simplify conduct of the experiment, and to have a more reasonable sample size, these students were both identified and left in the aggregate data below. Additionally, a prerequisite course currently incorporates some mention of this in a lecture or two. Even allowing for this prior awareness, the results show a general strengthening of student awareness.

In Figure 2 the students were polled before and after experiencing the colorblindness demonstration. After the demonstration the students were asked to rate on a scale of one to five the following questions: Their awareness of the need to consider colorblindness when designing software and similar systems. Their awareness of the need to consider the differently-abled in general. Their awareness of the need to consider the differently-abled when writing code and developing applications. After the demonstration they were also asked two additional questions regarding the experience, touching on the on their knowledge and awareness. A query on longer term impact ("do you think you'll remember") was also posed (see Table 1).

Table 1: Average Scores – Immersion and Recollection

Question	Average Score
Did the immersive experience in the VisCube visualization system increase your knowledge of the need to build software with this issue in mind?	3.79
Do you think you'll remember this experience long term?	4.00

For the first question, which asked them about their awareness of the need to consider colorblindness when designing software and similar systems, there was a 21% increase in awareness post demonstration. The second question,

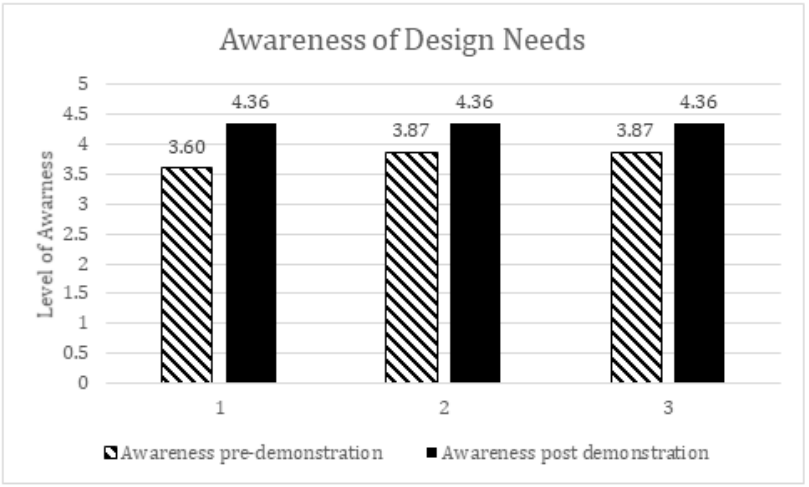


Figure 2: The three categories are the polled class's self-recorded awareness for the need to consider a topic. Categories 1's topic was 'colorblindness when designing software'. Categories 2's topic was 'Differently-abled in General'. Categories 3's topic was 'Differently-abled when building software applications'

which asked them about their awareness of the need to consider the differently-abled in general, had a 13% increase in awareness post demonstration. The third question asked them about their awareness of the need to consider the differently-abled when creating applications, also had a 13% increase, which may also reflect the similarity of the queries. The last two questions both had most of the participants reported that the VisCube colorblindness demonstration did increase their knowledge of the needs to build software systems with colorblindness needs in mind.

5 Conclusions and Future Work

The initial survey results indicate an overall improvement with awareness. This initial trial data indicates that a compelling environment was created. A longer term follow up, if only to test the duration of impact, is planned as part of the ongoing use of this classroom exercise as a part of the departmental long term assessment program. Further deployment of the exercise to other courses is also planned, including both the capstone project course and the initial sophomore course with a design focus. The classroom-friendly length

of the demonstration makes this particularly easy to implement this, but will generate a need for additional scenarios and environments. It is expected that this eventual ensemble of tools will broaden and deepen the impression the experience makes on students.

From a graphics and technical standpoint there are improvements that can be made, including increases to the overall the visual fidelity. The implementation of other forms of colorblindness, particularly tritanopia, is also planned for the near term. Context and supporting materials, including video from those with colorblindness and other ability issues, are also being investigated for use outside the classroom and to enable more reflection on this topic.

The Valparaiso Department of Psychology has also expressed in both the initial demonstration and in crafting similar tools to support their courses. Further interdisciplinary efforts are under consideration in partnership with the programs in the University's College of Nursing and Health Professions. It is expected that these efforts will also include trials with HMD display systems for greater portability and to test implementations with lower overhead costs and facilities requirements.

References

[1] Colour Blind Awareness Project. Types of color blindness. http://www.colourblindawareness.org/colour-blindness/types-of-colour-blindness/.

[2] Præcipio Consulting. Paying for mistakes: The cost to fix a software defect and how to avoid it. https://praecipio.com/pc/display/ae/2014/10/09/Paying+for+Mistakes%3A+The+Cost+to+Fix+a+Software+Defect+and+How+to+Avoid+It.

[3] ACM Computing Curricula Task Force (Ed.). Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science.

[4] Peter Freeman. Essential elements of software engineering education revisited.

[5] US Government. Americans with disabilities act 1990. https://www.ada.gov/pubs/adastatute08.htm.

[6] US Government. Section 508 homepage. https://www.section508.gov/.

[7] Fernanda Herrera, Jeremy Bailenson, Erika Weisz, Elise Ogle, and Jamil Zaki. Building long-term empathy: A large-scale comparison of traditional and virtual reality perspective-taking.

[8] National Eye Institute/National Institutes of Health. Facts about colorblindness. https://nei.nih.gov/health/color_blindness/facts_about.

[9] Stanford University Press Office. Virtual reality can help make people more compassionate compared to other media. https://www.eurekalert.org/pub_releases/2018-10/su-vrc101518.php.

[10] Kristen Shinohara, Saba Kawas, Andrew J Ko, and Richard E Ladner. Who teaches accessibility? *In Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18*.

[11] Craig Stephenson. The physics of color vision and color blindness. http://ffden-2.phys.uaf.edu/212_spring2005.web.dir/Craig_Stephenson/colorblindness.html.

[12] Unity3D. Unity documentation 2018.2 - LUTs. https://docs.unity3d.com/Manual/PostProcessing-UserLut.html.

[13] Unity3D. Unity documentation 2018.2 - post-processing stack. https://docs.unity3d.com/Manual/PostProcessing-Stack.html.

[14] US Census. Age and sex distribution in 2005. https://www.census.gov/population/pop-profile/dynamic/AgeSex.pdf.

[15] Austin van Loon, Jeremy Bailenson, Joshua Bostick Jamil Zaki, and Robb Willer. Virtual reality perspective-taking increases cognitive empathy for specific others.

[16] Visbox Inc. VisCube product description. http://www.visbox.com/products/.

[17] W3C. Japan - web accessibility initiative. https://www.w3.org/WAI/Policy/policy/japan/.

[18] W3C. Web accessibility laws policies. https://www.w3.org/WAI/Policy.

[19] Brian Wentz, Paul T Jaeger, and Jonathan Lazar. Retrofitting accessibility: The legal inequality of after-the-fact online access for persons with disabilities in the united states.

Getting Ahead with a Hat: Reengineering a Computer Organization Course*

Michael P. Rogers, Charles Hoot
Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
mprogers@mac.com, hoot@nwmissouri.edu

Abstract

This paper describes efforts to reengineer and revitalize a computer organization course to make it more relevant and more appealing to our students.

1 Introduction

For many years our university has offered a single, sophomore-level course in computer organization that covered an ambitious amount of material, from history of computing to building a simple CPU, with a large dollop of assembly language programming in between. The course was highly challenging and perceived by our students as being largely irrelevant.

Part of the problem arose because the course was originally two: one in computer organization, and one in computer architecture. When they were condensed into one course, the designers were reluctant to compromise coverage of important topics. However, as the field of computer science has continued to evolve, so have the ACM curricular guidelines to which we pay heed. It was recognized that significant changes had to be made.

This paper describes those changes, beginning with an overview of the old course, goals of the redesign, the technology choices that we made when revamping the course, and student perceptions of the result.

2 Overview of the Old Course

The original course, inherited by the authors, covered a wide swathe of topics, loosely adopted from the classic text by Patterson and Hennessy[5]: a history of computing; data representation; benchmarking; a considerable amount of MIPS assembly language programming up to recursive functions; digital logic design; building a basic ALU; the fetch-execute cycle; a MIPS data path; and memory.

The course was problematic for three reasons.

First, it was simply too ambitious for a one semester, 3 hour course, and was in need of judicious editing. It spent too much time covering topics that the ACM curricular guidelines now deem of lesser importance (e.g., digital logic design).

Secondly, the choice of assembly language, while it should be irrelevant, does affect the tone of the class. Teaching MIPS assembly on the MARS simulator simply did not resonate with our students. In spite of instructor protestations to the contrary, students felt that the course was entirely divorced from anything that they were likely to see in practice, and therefore irrelevant. From an instructional standpoint, finding useful instructional material, and resolving issues with MIPS was challenging, with searches leading to pages that were often a decade old.

Finally, the course represented missed opportunities: we had a number of gaps in our curriculum, critical topics that were either very lightly covered, or simply left to electives that some students would never take. These topics included C programming, as well as UNIX/Linux command-line tools and system administration. C, so close to the hardware that it has been described as a "machine independent assembly language"[2], is consistently the second most popular language on the Tiobe Index[6], and a good fit for a computer organization course. Knowledge of the UNIX/Linux toolkit is essential in so many areas — even Microsoft now officially supports GNU/Linux on Windows[1]. To omit it seemed to border on curricular malpractice and it became an essential part of our re-engineering strategy.

3 Goals of the Reengineering

The main goal of the redesign was to encourage a hands on exploration of the material through the judicious use of hardware, i.e., we wanted students to have a chance to play with hardware. A secondary goal was to give our students more exposure to command-line level tasks, and give them a peek at what goes on behind the scenes of an IDE. To accomplish these goals, the course now uses Raspberry Pi's as the basic platform. This gives the

students a chance to write and run real assembly language programs on a real machine, and use a debugger to view the contents of actual memory directly. The addition of a custom designed hat makes assembly language programming a more visceral and tactile experience. While not complicated, the hat has both input and output that serves as an introduction to the wider world of embedded programming.

3.1 The New Course

The course starts with general computer architecture, but quickly adds in the Raspberry Pi to reinforce theory with practical experience in ARM assembly. Midway through the course the hat is added to the mix to increase the wow factor and facilitate discussion of I/O beyond the keyboard and screen. In the final portion of the course, C is introduced to illustrate connections between higher level languages and assembly.

3.2 Course Topics and Coverage

The following is a detailed description of the new syllabus, with major topics bolded for easy reference.

1. **Base arithmetic and the representation of integer data types**. This is essential information that they can readily master and serves as a gentle introduction to the course.

2. **A general overview of a computer architecture**. The fetch-execute cycle is discussed using a simplified model of a cpu and assembly language.

3. **Welcome to the Raspberry Pi**. Students receive their Raspberry Pis and learn valuable skills that are not emphasized elsewhere in the curriculum. They start with the basics, installing the Raspian OS on an SD card, and learning how to connect to the Pi using either ssh or VNC. For many students, this is the first time that they have had a chance to work at the command line level in a UNIX environment. The students work with the nano text editor, giving them an opportunity to practice the basic file system Linux commands that they learn in this section.

4. **ARM assembly and more architecture**. Students are introduced to general architecture concepts such as memory, registers, assembly and machine code, using ARM as a concrete example. Fundamental assembly language operations, including data movement, branching, and basic math, are covered. Simple programs are created, assembled into machine

code and then run, using gcc throughout. This affords an opportunity to discuss the differences between machine code, assembly, and higher level languages.

5. **Debugging programs**. The gdb debugger is introduced so that students can debug their assembly language programs. The debugger is used throughout the remainder of course as a tool to explain and explore the structure of programs and data. This is a big advantage over the prior version of the course. Now students get to see real structure on a real device as opposed to just a simulator (MIPS and MARS).

6. **Pointers**.Students are introduced to pointers, implemented in ARM assembly using register indirect addressing mode. Since our students have typically used Java and Python prior to this course, pointers are a major step. The differences between Java references and assembly language pointers are explored, and we prepare for pointers in C later in the course. Strings are introduced as well.

7. **Memory hierarchy**. Caching and the memory hierarchy are discussed along with their effects on program performance. As the course is currently configured, the discussion of performance is theoretical only with no concrete examples.

8. **RISC vs. CISC**. A theoretical discussion of the differences and relative performance of the two architectures is presented. ARM is used as an illustration of RISC. This leads to a low level discussion of the structure of machine code instructions.

9. **Branching**. Conditions at the assembly language level are introduced and contrasted with the abstractions of a higher level language. The connection between assembly and higher level languages is further strengthened by showing how a general if-then-else construct translates into assembly language.

10. **Low level input/output**: The General Purpose Input/Output (GPIO) pins on the Raspberry Pi are briefly described. We give the students a small taste of electronics, but recognize that we are not teaching computer engineering. The goal is to give them enough information so that they can understand conceptually what is happening with the components on the Pi hat when they call functions in the Wiring Pi library. To start, students will control the three LEDs and then later work with the buzzer and switch. Polling is discussed in this unit.

11. **Looping**. Looping in assembly is addressed and contrasted with a general while loop in a higher level language. Now they can make the LEDs blink!

12. **Character I/O**. While students have already been exposed to the printf() function for doing character-based output, a more substantial exploration is undertaken.

13. **Functions**: To this point, students have been calling pre-existing functions with a limited number of parameters. The full story of calling conventions and using the stack to pass parameters and return values is presented as students write their own functions.

14. **Arrays**: Students get a more detailed view of how an array of data is stored in memory. Accessing arrays using a base and offset motivates the use of the more advanced addressing modes provided by ARM.

15. **Memory use**: The typical partition of memory into code, data, stack and heap areas is presented and the difference between global and local variables is highlighted.

16. **Pipelining**: This is primarily a theoretical discussion, though ARM is used as an example. Assembly code reorganization to avoid hazards and stalls is shown along with the information that modern compilers will attempt to optimize their code to keep pipelines full.

17. **Floating point**. The IEEE 754 standard for floating point is shown. Normalization and the special NaN codes are discussed. Finally, the details of how floating point values are stored in registers/memory and the assembly language operations available on ARM are shown.

18. **An introduction to C**. Even though C is a high level language it is closer to assembly than many other languages and is relatively simple. About 4 weeks are taken to present the core of the C language. Using gdb, the students can tie the C code back to the assembly and see how much the higher level language is abstracting away. We also go out on a high note since working with C is easier than working in assembly.

4 Building a Hat

In this section we outline the steps required to design, build and deploy a hat, and tips on how to integrate it into the curriculum.

Our first step was to ask ourselves whether, in fact, we needed to build a hat. Could we just choose from several already on the market, or take another tack

and have students construct circuits on a breadboard? We opted for a custom build for several reasons. We were on a budget, and could fabricate a hat with just the components we needed for less than commercially available products. As a side benefit, it provided the opportunity to familiarize ourselves, as well as a small group of student assistants, with the hardware development lifecycle, and hone skills that are useful in another class that we offer, the Internet of Things (IoT).

In contrast, letting the students prototype on a breadboard would detract from the main goals of the course, consume too much class time, and add uncertainty when trying to debug code. Furthermore, in our IoT course, students do prototyping for much of the semester, so students interested in working with hardware directly already have an option.

Instructors acquainted with Arduino microcontrollers will feel right at home with the Raspberry Pi. The most recent models, based on the Broadcom's BCM 2837 system-on-a-chip, expose 26 GPIO pins that can be programmed using the WiringPi[3] library. The operation — set the pin mode and then write or read values — is the same, and the WiringPi library uses familiar method names (See Table 1). While WiringPi is sizable, these 6 methods are all that are required to create a wide range of intriguing assignments.

Table 1: WiringPi Library Methods

Method	**Description**
void wiringPiSetupGpio();	Initializes the WiringPi library
void pinMode(int pin, int mode);	Establishes whether a pin will be used for input, digital output, or PWM (Pulse Width Modulation) output.
void digitalWrite(int pin, int value);	Turns the pin on (value = HIGH) or off (value = LOW). We can use this to turn LEDs on and off.
void pwmWrite(int pin, int value);	Sets the duty cycle of a pin configured for PWM (value ranges from 0 (0%) to 1024 (100%). We can use this to control LED brightness.
int digitalRead(int pin);	Reads the voltage presented to a pin and returns either LOW or HIGH. We use this to determine if a button is pressed.
int softToneWrite(int pin, int freq);	Causes a pin to emit a square wave of a given frequency. By connecting this to a buzzer we can play music.

Version 1.0 of our hat design uses 3 LEDs, a switch and a piezoelectric buzzer: we wanted to start small, and learn from our inevitable mis-

takes. The basic electronics are quite straightforward [2, 4]. Basically, when `digitalWrite(pin,HIGH);` is executed, a voltage is applied to the pin; it flows through an LED back to ground (another pin), completing the circuit so that the LED glows. The `softToneWrite()` function behaves similarly with respect to the buzzer, except the voltage fluctuates, generating a tone. In the case of the switch, when it is open a voltage is applied to the connected pin, so `digitalRead()` returns `HIGH`; when it is closed, the voltage drops to 0, and `digitalRead()` returns `LOW`.

We began with a rudimentary prototype, connecting a 40-pin ribbon cable to our Raspberry Pi, and jumper wires from the cable to our prototype board (see Figure 1).

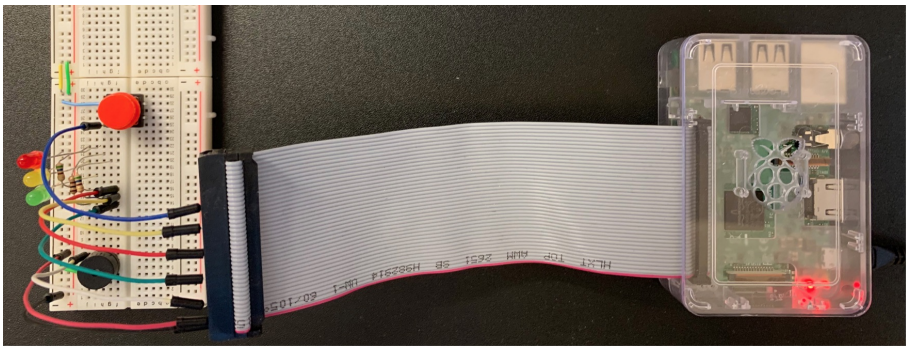


Figure 1: The Hat Prototype

Once we were sure our circuit functioned, we transformed our schematic into a printed circuit board using DipTrace, a CAD program that is popular among hobbyists (see Figure 2).

The boards were produced by OshPark. They offered prototypes at $5 per square inch for 3 boards, and shipped within 12 calendar days. The cost of a medium run was $1 per square inch with a 100 square inch minimum shipping within 15 calendar days. Frugality was an important consideration throughout the design process. The total cost of both Raspberry Pi and hat came to $63.13, per student which is quite affordable. To reduce the instructor's responsibilities, the distribution of the hardware was delegated to the library: students would check out the equipment for the semester. This had the added benefit of a standardized method for charging students if they did not return the Pi, or returned it in a damaged state.

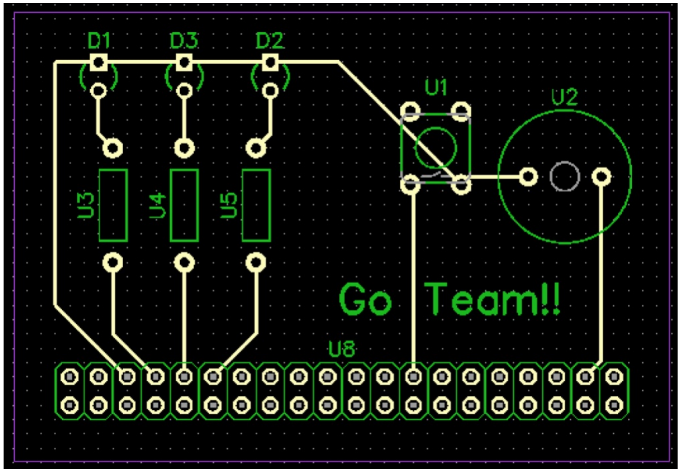


Figure 2: Hat Printed Circuit Board

5 Student Feedback

We did do a pre- and post-course survey to evaluate general knowledge of course topics (Table 2). The results are decidedly mixed, and not as high as we would like, but understandable considering when the survey was administered – towards the end of the semester but before students had begun to review material for the final exam. We anticipate that these numbers will improve considerably when we teach the course for a second time.

Since the vast majority of students were taking this course for the first time, a formal survey, comparing their current experience to that using MIPS and MARS, was not practical. But, based on post-course comments and student interviews, students were generally pleased with the reworking of the course. While students felt that the course was challenging, they also thought it was interesting and enjoyed working with the Raspberry Pi and the hat. They also said that it prepared them well for later courses. Anecdotal evidence from one student who was repeating the course (with the same instructor) indicated that they definitely preferred the redo to the original, and that it made more connections between computer organization and the rest of the curriculum.

6 Conclusions

Reengineering the course has definitely made it more relevant and engaging. By incorporating affordable hardware that only requires a small number of

Table 2: Pre- and Post-Course Survey Results

Question	**% Correct, Pre Course**	**% Correct, Post Course**
How many values can a bit hold?	36.7	55.3
How many bits can a byte hold?	73.3	75.7
How many values can a byte hold?	40.7	75.7
What register contains the address of the instruction about to execute?	13.3	73.0
Where should you store a data value that is going to be used often in a program?	43.3	70.3
Which of the following architectures has simple instructions?	27.1	77.8
Purpose of pipelining	23.7	62.2
The code that runs on a CPU is	56.7	73.0
When you cannot find a data value in cache, what is that called?	35.6	64.9
Given int r [] = {10, 20, 30, 40}; what is *(r+2)?	45.0	63.9
The program that translates assembly into machine code	31.7	62.2

library functions, students are better motivated and able to grasp lower level architectural details. This allows weaving in abstract topics while providing opportunities to write tangibly interactive assembly language programs. It also facilitates the transition from assembly to the higher level language C where they can appreciate the economy of code that abstraction allows. As a side benefit, the course fits better in our curriculum. The combination of hardware, assembly and C leads naturally into our IoT and OS courses.

References

[1] Microsoft Corp. Learn about the windows subsystem for linux — microsoft docs. `https://docs.microsoft.com/en-us/windows/wsl/about`. Retrieved on October, 20 2018.

[2] Harry Fairhead. *Raspberry Pi IoT in C*. IO Press, 2016.

[3] Gordon Henderson. Wiring pi. `https://wiringPi.com`. Retrieved on October, 20 2018.

[4] Simon Monk. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions.* OREILLY, 3rd edition, 2019.

[5] Patterson and Hennessey. *Computer Organization and Design.* Elsevier, 5th edition, 2014.

[6] tiobe.com. Knuth: Computers and typesetting. `https://tiobe.com/tiobe-index/`. Retrieved on October, 20 2018.

When the Play Is "The Thing" and Not the Software: Student Experiences Engineering Software for a Theatre Production *

Brian Kokensparger
Journalism, Media and Computing Department
Creighton University
Omaha, NE 68178
402 280-2878
bkoken@creighton.edu

Abstract

In Fall 2018, a software engineering class collaborated with Creighton University's theatre program to produce software extending the production of a new play that was partly set in the "twittersphere". Employing an Extreme Programming approach, the students discovered that the theatrical staging process had a lot to say about the software engineering process, especially in the employment of "soft skills" surrounding requirements elicitation and user interface development. This paper presents themes drawn from students' reflections as well as those from the course instructor, all of whom call for a more structured planning process and special attention to change and time management during the development period. The instructor also reflects on the question, "How is developing software for an actual theatre production a good learning experience for software engineering students?"

1 Introduction

A software engineering course has as one of its chief outcomes that students gain experience with software evolution, including change management, estimating the impact of changes, and the task of refactoring [1]. Therefore, course instructors often look for "real world" opportunities for students, where they work on projects that are likely to evolve both intrinsically and extrinsically, and yet are also too embedded in the real world to fail.

In addition to these real-world requirements, this also involves working with "real world" students, who have a number of other courses to manage at the same time, a social life, trips home, and other things that workers in industry do not usually encounter. And instead of offering a paycheck and continued employment as carrots, the instructor only has the end of semester grade (and perhaps aspirations of a career in the field) to use as a measuring stick and motivator.

The literature provides some support for using software to support theatre productions in higher education [2, 4, 3]. In the course instructor's former experience doing college-level theatre, he discovered that it has a specific purpose – the goal of training actors and crew members – and often disregards commercial appeal to educate the audience and actors alike. This results in the theater producing plays that may not be performed regularly by local professional and community theaters. In this educational context, there is a certain amount of latitude given to the performers and stagers of the play, and in turn the audience does not expect high-budget commercial production values.

Apply this to projects in a college software engineering course (versus software engineering in industry): The college developers are free to try new things, to have the little approximations and micro-failures that one expects from student work, and to produce a best effort deliverable based on limited resources (in terms of time, budget, available equipment, and support).

Yet, in theatre, when the curtain opens, the show must be ready to go on. There are no exceptions to this reality. Theatre is theatre, and no matter where or how it is done, when opening day arrives, the curtain opens, and human beings must be ready to perform. If they are not prepared for this reality, then it may be a painful experience for everyone involved. But the curtain still opens.

It is the resolution of these two opposites that challenged the SE course instructor to work in a context where micro-failures are accepted and even expected, yet also to work in a context where a curtain literally "opens" on the project, requiring the students to deal with that reality.

Within this context, this project considered the question: "How is developing software for an actual theatre production a good learning experience for software engineering students?"

2 Context of The Project

2.1 The Script and the Production

Creighton University's theatre program chose a new script written by an alumna as a world premiere of the play. The script was billed as partially taking place in the "twittersphere," and incorporated human actors both as onstage characters and their avatars. In the premiere production, the director decided to stage the onstage characters and avatars as different actors and direct the avatars to hang in their own space on the edges of the stage, while the human characters used the space in center stage to interact with each other, as well as with the props and set features.

As the script had a technical theme, it was a perfect first project for the instructor to develop a partnership with the play's director, and to work out the ways and means of mounting both a production of the play and development of associated software. The playwright was in residence for the last month of rehearsals and for performances, which allowed actors and the software engineering students direct access to her for questions and other support.

2.2 Overview of Extreme Programming

Though Extreme Programming (XP) is difficult to define, it began in 1996 as a variation within the agile approach and emphasizes a customer-centric approach to software development that responds quickly and effectively to changes and emphasizes teamwork, respect, and courage in the development cycle [5].

As XP utilizes User Stories to elicit requirements, this focus on the user experience and expected interactions with the software requirements in more of a narrative format seemed a perfect approach for working with a play director, who is immersed in story and extended sequential interactions.

Of additional significance in terms of the XP approach to the project was the dialogical dynamics of the process. These dynamics are presented below.

2.3 The Software Engineering Project

To meet the student learning outcomes identified above, the software engineering project associated with the theatre production involved the development and delivery of software applications that the cast, crew, director, playwright, and audience could use before and after the performances to interact with one another, and to engage the production in a deeper way. The forms that these applications took were determined through the XP practice of articulating User Stories (a customer task), estimating the required resources to actualize the

User Stories in software (a developer task), prioritizing the User Stories (a customer task), developing the software applications by priority (a developer task), user testing the software to provide feedback (a customer task), and doing final debugging and revision to produce a deliverable (a developer task), on or before the articulated deadline (set by the customer).

3 Project Methodology

It was determined by the customer (the play's director), that the software development would be focused on enhancing the audience's engagement with the play through online applications that could be accessed before and after performances. No software would be developed for use during the actual performance itself. This allowed for the play to be offered in a traditional format, without the potential obtrusion of technology. In XP, this is solely a customer decision; the developers had no vote on the matter but agreed with the decision.

The entire software engineering class (the developers) met directly with the director (the customer), prompting her to articulate User Stories and asking pertinent follow-up questions to make sure they understood the details.

After the meeting, the students wrote a short collaborative document articulating in as much detail as possible what they heard from the User Stories session. After discussion, this collaborative document was revised until it reflected the consensus of the students who attended the meeting.

Once the User Stories were documented, the students met to estimate the time and required resources (target device, platform, development language, etc.) for each of the User Stories. The instructor then took these estimates back to the director, who prioritized them. Afterwards, he documented these priorities and presented them to the students. Due to limitations in space, the stories and elicited functional and nonfunctional requirements could not be provided in this paper.

The students were then assigned to two groups: one group developing applications facilitating the interactions between the actors (as themselves) and the audience members, and another group developing applications facilitating the interactions between the characters and the audience members.

After the groups were assigned and given their prioritized User Stories, they were directed to meet and distribute individual tasks using a Pairs Programming paradigm, which is an important feature of XP. Development iteration planning meetings were held during class sessions, with both groups meeting in separate corners of the classroom, employing the XP practice of All Engineers in One Room.

Near the end of the project period (ten days before opening night for the production), both groups merged into one group to work on those areas where

the software applications needed to interact (for implementation of a shared password, development of a landing page to allow access to all of the software applications, and adjusting the look and feel of the applications to a single design standard).

After user acceptance testing (where the instructor stood in for the director, who was understandably too busy with technical rehearsals and last minute details to test the applications herself), the URL for the landing page and password were released to the theater box office for distribution, thus signaling the software release for free access by audience members, cast, crew, the director, and the playwright.

4 Student and Instructor Experiences

4.1 Student Experiences

Immediately after the project was completed and released for public access, a survey was administered, as a self-evaluation of each student's work on the project, as well as a reflection over what they learned, what they discovered they are "good at" (strengths), and what they discovered they need to learn more about (challenges). There were 12 students enrolled in the course, and all 12 students returned the surveys. These surveys consisted of open-ended questions, designed to elicit themes under qualitative analysis.

4.2 Student Survey Results

In the initial survey results, there were great variations among the individual responses. Some students, for example, cited design skills as a strength, while others cited design skills as a need for further learning. These specific skill-level responses were likely related to the nature of the project itself. Since this project required significant backend development, that reality likely inspired those varied responses. Therefore, the skills responses to the survey basically canceled each other out, and were not helpful in addressing the research question.

The software engineering instructor was more interested in the soft skills – teamwork, planning, project design (versus UI design), leadership, and time management. Using qualitative analysis techniques, some themes were generated from the student responses, and are provided in Table 1.

These themes suggested a difference between doing a software engineering project within a controlled environment (where the customers and most other stakeholders are technically knowledgeable), and in an open and uncontrolled environment of technical laypersons, such as a theatre production. Notably:

Table 1: Soft-skill Themes from the Survey of Software Engineering Students Regarding Their Project Experiences

Things I Learned	**Strengths**	**Challenges**
Dealing with change and expectations	Debugging code when under pressure.	Time management
Importance of having a project manager	Discovered I am a good writer!	Communicating with non-CS people.
Importance of teamwork	Planning and collaboration in the planning process	A good version control tool

The "Things I Learned" themes revolved around the student learning outcomes for the course, which centered upon change management, project management, and group work. From the survey, and the software that the students produced, it was clear that students were becoming proficient in these specific learning outcome areas.

The "Strengths" and "Weaknesses" themes reinforced the learning outcomes, including working under time constraints, the importance of written and oral communication with non-technical users, working with others, and version control. These strengths and weaknesses were particularly enhanced by the domain of working with a theatre production. The pressures of time management were exacerbated by the project: When opening night arrived, it arrived. When the show opened, the software had to be up and accessible, and it had to work. This was the ultimate reality. Several times, when working in class, students commented about how stressful the experience was. It is a future research topic to determine if the stress was "good stress" or "bad stress," but the reality was that having opening day loom large before the class tempered their resolve in getting the software completed and functional for non-technical users.

4.3 Instructor Experiences

Although it is an instructor's job to be proactive – to anticipate stumbling blocks for students and to try to eliminate them – for this project, the instructor resolved to let the students experience a real project in the real world: Complete with its micro-successes and micro-failures. They were free to make their own choices and live with them. This is the most unique benefit of working with a theatre production. As the play director cannot get up on stage and do the acting for the cast on opening night, the instructor could not make the choices and write the code and test it for the students. It was their "show," and all he could do was try to prepare them and counsel them towards making

good choices that led to success.

Mistakes were inevitable, and a couple occurred with this project as well:

- They missed the stated one week before opening deadline for release, and the applications ended up being released six days later – the day before opening night.
- The original goal and design was to introduce the audience members to all of the characters in the play but withhold one character (who would have been a "spoiler" if included in pre-performance accesses) until after the user viewed the performance. The instructor was surprised to hear one of the students say "just got an email from the director, and we don't have to worry about locking and unlocking the spoiler." This was a significant change from the User Story, that had specified that the spoiler character would "unlock" only after the users viewed the performance. Important changes like this one should have been submitted to the dialogical processes set out in the XP approach to software engineering.

In an educational setting, the presence of these micro-failures gave the class an opportunity for discussion and learning. Just as an actor on stage will flub a line, or occasionally miss an entrance, software engineering students will also make mistakes. Better they make them in the classroom and within the safe confines of an educational community than out in industry, where the stakes are higher.

Does a software engineer in the corporate world lose her job over missing a deadline? It's been known to happen. In a college software engineering class, a student might get a C on the project instead of an A or a B. The consequence for this micro-failure is not nearly as high stakes for students (but it still hurts).

5 Conclusion

As the research question for this project consisted of "How is developing software for an actual theatre production a good learning experience for software engineering students?" the results of this qualitative study are that having software engineering students collaborate with actors and directors in theatre productions provides real-life production experiences with immovable deadlines, but also introduces soft skill challenges that are not typically encountered in industry, such as working with non-technical customers, motivating customers who have other goals besides software production, and dealing with significant changes to project requirements and deliverables right up to release date. This provides software engineering students with learning opportunities that are difficult to get in typical classroom projects, and other areas where the software is "the thing" in which everyone is focused. When the play itself is

"the thing," the software becomes "that other thing," and this difference provides an excellent learning opportunity for students to develop their soft skills (along with the micro-failures that come with it), intensified by the context of "too embedded in the world of real people" to fail.

References

[1] ACM IEEE-CS Joint Task Force. Computer science curricula 2013: Final report. `http://ai.stanford.edu/users/sahami/CS2013/final-draft/CS2013-final-report.pdf/`. 2013. Retrieved November 24, 2017.

[2] Stephan Krusche, Dora Dzvonyar, Han Xu, and Bernd Bruegge. Software theater - teaching demo-oriented prototyping. *ACM Transactions on Computing Education (TOCE) - Special Issue on Capstone Projects*, 18(2), July 2018.

[3] Michael Skirpan, Jacqueline Cameron, and Tom Yeh. More than a show: Using personalized immersive theater to educate and engage the public in technology ethics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, 2018. ACM.

[4] Michael Skirpan, Jacqueline Cameron, and Tom Yeh. Quantified self: An interdisciplinary immersive theater project supporting a collaborative learning environment for cs ethics. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 946–951, New York, NY, USA, 2018. ACM.

[5] Don Wells. Extreme Programming: A gentle introduction. `http://www.extremeprogramming.org/`. Retrieved November 13, 2018.

Teaching AP-CSP In A College Setting Using An AP-CSP Endorsed Curriculum: An Experience Report*

Tim DeClue
Department of Computer and Information Sciences
Southwest Baptist University
Bolivar, MO 65613
tdeclue@sbuniv.edu

Abstract

The College Board's AP-Computer Science Principles (AP-CSP) course is a recent development in advanced placement courses and is significant due to its explosive growth and widespread curriculum support in the United States. This paper discusses the AP-CSP course, describes the author's experience teaching the course to traditional college students in a university setting, and suggests some changes that may be worth noting should others wish to replicate the experience.

1 Introduction

In May of 2016, the College Board began testing the Advanced Placement Computer Science Principles (AP-CSP) course for the first time following several years of development. According to the College Board, this course "... is designed to be equivalent to a first-semester introductory college computing course" [3, p.4]. AP-CSP joined APCS-A, a course supported for many years and focused on Java-based programming, as the two advanced placement computer science courses supported by the College Board. In the fall of 2017, high school students began matriculating to college and receiving college credit for

their advanced placement work. In the spring of 2018, the author began providing professional development to K12 computer science teachers using the Code.org curriculum. In the fall of 2018, the author taught the university course for which credit is granted for AP-CSP, and the author chose to use the Code.org curriculum to support the university course.

2 Course History and Content

AP-CSP was developed over several years by the College Board in partnership with the National Science Foundation, a variety of secondary and post-secondary education providers, and computer science professional organizations[3]. The course is designed to appeal to a broad range of students, particularly underrepresented groups, and is intended to be both relevant and have content integrity with classical computer science. Ten different organizations are endorsed by the College Board to provide curricula and professional development which supports teaching AP-CSP [2]. Some endorsed content providers, like Code.org, stress making the course accessible to all high school students and explicitly seek to avoid advantaging students with prior computing experience or background. According to the College Board, "students will develop computational thinking skills vital for success across all disciplines, such using computational tools to analyze and study data and working with large data sets to analyze, visualize, and draw conclusions from trends." [3, p.4]. The course content is built upon the intersection of six computational thinking practices with seven big ideas in computer science as shown below.

CSP Big Ideas	**CSP Computational Thinking Practices**
1. Creativity	1. Connecting computing to real life
2. Abstraction	2. Develop programs
3. Data and Information	3. Abstracting to break down problems
4. Algorithms	4. Analyzing problems
5. Programming	5. Communicate ideas orally and in writing
6. The Internet	6. Collaborate with others to solve problems
7. Global Impact	

In a break from traditional AP courses, AP-CSP not only tests knowledge of the content with a test at the end of the school year, but also requires two non-trivial applied projects be completed and submitted for assessment by the students during the course. These two projects — called the Explore Task and the Create Task — account for 40% of the AP exam score. The Explore Task is a research activity which culminates in a creative digital artifact, and

the Create Task is a creative activity which culminates in a piece of software designed and programmed by the student and a partner.

These ideas, practices, and tasks are combined with rigorous, relevant computer science content to form a significant contribution to the student's computing education. Students leave the course with a solid computer science foundation able to effectively inform their daily life.

3 The AP-CSP Code.org Curriculum

Code.org implements the big ideas and practices in computing in a web-based curriculum structure, a set of research-based instructional strategies, a variety of resources and tools, and both web-based and face-to-face professional development for K12 teachers [4]. Student activities include a mix of "plugged" and "unplugged" learning.

The Code.org curriculum divides the content into five units.

Unit	**Description**
Unit 1: The Internet	Illustrates how the internet is multi-layered and explores interesting aspects of the internet including issues related to reliability and security.
Unit 2: Digital Information	This unit focuses on how computers store complex information using binary approaches.
Unit 3: Programming	The Code.org curriculum uses JavaScript to illustrate problem-solving through the use of programming.
Unit 4: Big Data and Privacy	This unit explores the intersection of current events, policy, law and ethics as it related to data and encryption.
Unit 5: Building Apps	Students are provided additional opportunities to explore programming approaches in Code.org's App Lab.

Woven through the units are activities which support and lead to the Explore Task and Create Task through-course assessments.

4 The Experience Report

The author used the Code.org supplied curriculum with only three deviations from the Code.org curriculum guide. These deviations were related to the schedule, the introductory programming environment, and the choice not to

have the students take the advanced placement test at the end of the course. The deviations are explained below.

4.1 Schedule

The university setting where the course was taught utilizes a traditional fall/spring sixteen-week semester schedule and utilizes the last week as a final exam week. The AP-CSP is designed to be delivered over an entire school year with the AP test taking place in May. Due to this one semester constraint, the author divided the course into segments which could fit into a single sixteen-week semester as shown in Figure 1. Additionally, the course where the AP-CSP curriculum was implemented was a three-credit hour course which utilized two days of lecture and two days of lab each lasting 50 minutes apiece.

4.2 Performance Tasks

According to the Course and Exam Description for the AP-CSP course [3], students working on the Explore and Create Tasks are to be given a minimum of 20 class hours to work on these projects (8 hours for the Explore Task and 12 hours for the Create Task). This amount of time—equivalent to one-third of a single-semester course—was not a feasible use of time. Additionally, one reason class time is provided to high school students is so the teacher can make sure the students are the author of the projects they ultimately turn in. This issue is less of an issue for traditional college students as they are generally already living away from home and parents are usually less of a factor.

4.3 Programming Environment and Schedule

The Code.org curriculum has a built-in programming environment called App Lab [4]. App Lab is an effective environment for teaching programming at both the high school and college level, however, the context of the programming lessons is drawing and manipulating somewhat cartoonish ocean scenes. For this reason, the author chose to use Earsketch [1] — a different programming environment from another endorsed curriculum provider which utilizes music sampling and a built-in audio workbench—as the environment for the class the author taught at the college level. Earsketch was able to support all of the content included in the Code.org AP-CSP curriculum, but also allowed college-age students to create musical compositions using musical styles popular with the students.

JavasScript was used as the language, which corresponds to the language utilized in the Code.org curriculum, and the same lesson content was taught, although some choices had to be made regarding how much emphasis was

Weeks	Monday	Tuesday	Wednesday	Thursday
Week 1	**Unit 1 – The Internet** Syllabus, Course overview U1L1: Personal Innovations	Create Code.org account U1L2: Binary Messages	U1L4: Number Systems U1L5: Binary Numbers & the FlippyDo	U1L3: Sending Binary Messages with the Internet Simulator
Week 2	Binary numbers & ASCII codes U1L6: Sending Numbers	U1L7: Sending Text	U1L8: The Internet is for everyone U1L9: Battleship protocol	U1L9: Battleship protocol continued
Week 3	Labor Day – no class	U1L10: Routers & redundancy	U1L11: Packets & making a reliable internet	U1L12: The Need for DNS
Week 4	U1L13: HTTP & Practice PT Assignment	U1L14: Abstraction	Performance Task Presentations Test Review	**Test 1**
Week 5	**Unit 2 – Digital Information** U2L1: Bytes and File Sizes	U2L2: Text Compression	U2L3: Encoding BW images	U2L4: Encoding Color images
Week 6	U2L5: Lossy vs Lossless Compression	U2L6: Rapid Research	**Midterm Review**	U2L6: Rapid Research Showdown
Week 7	**Midterm exam**	Explore Task Workday	**Unit 3—Programming** U3L1: Programming Languages	Explore Task Workday
Week 8	U3L2: The Need for Algorithms	U3L3: Creativity in Algorithms	U3L4: Using Simple Commands	Fall Break - No Class
Week 9	U3L5: Creating Functions	U3L5: Creating Functions	U3L6: Functions and Top-down Design	Explore Task Presentations
Week 10	U3L7: Using API's and parameters	U3L8: Creating functions with parameters and return values	U3L9: Booleans and Random Numbers	U3L10: Looping and Random Numbers
Week 11	U5: Introduction to lists	U5: Lists and looping	U5: Recursion	U5: Building a Game (Rock, Paper, Scissors)
Week 12	U4L1: Big Data	U4L2: Visualizations	Review for Test	**Test 3**
Week 13	Debrief Test 3 and Create Task	Assigned U4L3: Check your Assumptions	U4L4: Rapid Research Data Innovations	U4L5:Identifying people with Data
Week 14	U4L6: The Cost of Free	Create Task Workday	U4L7: Simple Encryption	Create Task Workday
Week 15	U4L8: Encryption with Keys and Passwords	Create Task Workday	U4L9: Public Key Cryptography	Create Task Workday
Week 16	Review for the Final Exam Create Task Due			

Figure 1: Class Schedule

provided to programming topics. Finally, the Code.org curriculum teaches the programming content in two units; the programming content in the author's course was taught over a four-week period as one unit.

4.4 The AP Exam

The final deviation was simply that the students in the author's course did not take the AP exam. The reason was simply that the students were already receiving college credit for the course, so the need to gain AP credit for the course did not exist.

5 Conclusions and Observations

A first important conclusion was that the AP-CSP course content was very appropriate, relevant, and useful for college-age students. AP-CSP is billed as a college-level course and the author's experience supported this assertion.

A second conclusion was that the unplugged and plugged lab-based lab work provided by Code.org supported learning effectively. Students remained on task during class time each day during the semester. The unplugged activities, in particular were interesting and useful for the students. In one sense, the activities — due to their lack of reliance on technology — were refreshing and innovative to the students.

The last conclusion was that the Code.org instructional strategies were effective in supporting learning. In the author's opinion, these strategies — in combination with the innovative packaging of the course content — lie at the heart of the success of the course. There are six instructional strategies advocated by Code.org. These strategies are shown and defined below.

Strategy	**Description**
Lead Learner	A teacher-focused strategy which recognizes a shift from the teacher being the source of knowledge to the teacher being the leader in seeking knowledge.
Journaling	An opportunity for students to reflect on and write about an experience. Journaling can be digital or take place with physical notebooks.
Think-Pair-Share	A discussion seeding practice. Students are asked to think individually about a problem or task, then paired to explore the problem or task, and finally to share out what the pairs talked about.
Peer Feedback	The practice of students sharing their work with one another to prompt discussion, suggestions and iteratively improve their work.
Pair Programming	The technique of two programmers working together on one computer. The programmers have different roles named "driver" and "navigator".
Debugging	The practice of finding and fixing problems. Debugging is presented as a natural part of any creative activity which utilizes computers.

These six strategies promoted an active, engaged classroom which supported effective learning. Of even greater impact is the observation that these

strategies can be applied in other contexts and other courses with similar positive benefits. For example, during the semester the author was teaching this course, the author also implemented many of these same instructional techniques in a systems analysis and design course with similar results.

References

[1] EarSketch programming environment. https://earsketch.gatech.edu/earsketch2/.

[2] College Board. AP Computer Science principles: Adopt ready-to-use curricula. https://apcentral.collegeboard.org/courses/ap-computer-science-principles/classroom-resources/curricula-pedagogical-support. Retrieved November 25, 2018.

[3] College Board. AP Computer Science principles course and exam description updated fall 2017. https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf.

[4] Code.org. Computer science principles curriculum guide. https://curriculum.code.org/csp-18/.

Capstone as Consulting*

Denise M. Case, Charles Hoot
School of Computer Science and Information Science
Northwest Missouri State University
Maryville, MO 64468
{case,hoot}@nwmissouri.edu

Abstract

This paper describes the first semester in the restructuring of a two semester capstone course to frame project management, design, and implementation as a consulting effort. We draw upon real-world experience to introduce appropriate artifacts and procedures to the course. As part of profession-based learning, students will use industry-standard tools as recommended by a professional advisory team. This includes the introduction of JIRA and the expanded use of Git. Students learn project management tools and techniques early in the course and are then granted additional permissions to plan and implement under their own management and organization. The intent is to help students transition from novice developers to peer- and mentor-evaluated professionals that have developed an understanding of client perceptions and priorities, along with project scope, budget, and risk evaluation. As part of the initial phase of the project, students implement a variety of architectural prototypes which may include mobile and progressive apps.

1 Introduction

Common to many computer science programs, we have a capstone course where students implement a team project [7]. While most the originate from clients within the university (including faculty, staff, and students), some clients are members of the local community or are located remotely. Our capstone course

spans two semesters; the first semester traditionally focuses on design and the second semester focuses on implementation.

Creating useful software artifacts for a capstone project can be challenging for a variety of reasons. In particular, both clients and students in the course have varying capabilities. Client often have little understanding of software engineering and the development process. Students often have little understanding of interacting with a client or with team members. Added to this is the challenge of assessing the learning, competencies, and contributions of individual students within the larger group.

Motivation for this effort was four-fold: (1) better align student outcomes to industry practices, (2) adjust assessments to better reflect the tangible value produced by the individual and the team, (3) improve student interactions with a client, and (4) help students better understand professional obligations and the value of their work. In this context, with this motivation, we developed the following proposal: Use a structured consultancy model supported by industry-standard version control and project management tools. Initial scaffolds will be provided to guide the project in the early phases and help manage client interactions and expectations. This gives the appropriate foundations leading to student-led management of projects. Our work will provide a reusable, authentic approach to implementing this critical course.

2 Related Work

The course design builds on a variety of prior work from academia and industry. The Service Learning Practicum (SLP) model has been used successfully for two-semester capstone courses [2] and Carnegie Mellon has used a systems engineering consulting model in hundreds of partnerships over 18 years [8]. Both apply a consulting model to the course to create mutually-beneficial partnerships with non-profits to engage and motivate the students.

In 2009, Hurtig and Estell proposed a common framework for diverse capstone courses based on common project management phases of conceptual design and feasibility, design and definition, implementation, start-up, and project closeout [6]. The project-based approach employs a project review board (PRB) to guide and assess projects against standard rubrics [6].

The MVP schedule developed for this consultancy model, shown in Fig 1, was also influenced by the capstone framework timeline provided [6].

Clear, et.al, define terms and discuss a variety of issues related to project sponsors and issues that may arise [4]. Their work has influenced the concepts of roles presented in the next section. In the SIGCSE 2002 paper, Chamillard and Braun discuss course structure and specifically, discusses the important tradeoff between the balance between the development process and work

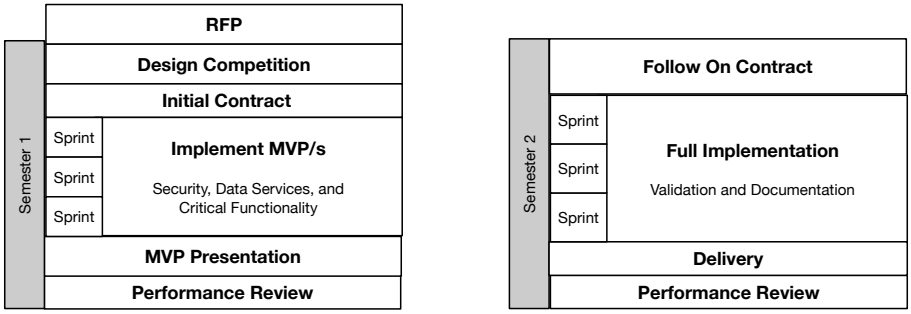


Figure 1: Consultancy Model Timeline

products created by the students [3]. They also describe the importance of a knowledgeable customer and the benefits this offers the students. They have students change teams, providing critical and inherent peer review that occurs when using artifacts developed by others. In a 1997 work, Dutson suggests possible course requirements and nine possible causes of team dysfunction, and suggests specific criteria for evaluation of the instructor, projects, classmates, and course [5]. In 2014, Acheson discussed the use of capstone projects sourced from industry partners and discusses interventions for problems that can arise [1]. A key outcome of the proposed model is the focus on applying the capstone course in a way that enables students to practice and demonstrate elements and outcomes from seven required enabling courses [10].

3 Terms and definitions

3.1 Consultancy Specific Artifacts

Request For Proposal (RFP): An RFP outlines basic requirements for a project and invites multiple entities to propose competing solutions. One or more solutions are selected to move into a contract phase. In our context, the competition is replaced by cooperative melding of designs into one initial contract.

Contract: A contract is the basis for the development and specifies the requirements that must be satisfied for a successful delivery. Each semester in the course has its own contract specifying the product to be delivered. As the first semester progresses, client feature requests are collected and prioritized for inclusion in the second semester contract for the completed application.

Minimally Viable Prototype (MVP): The endpoint deliverable of the first semester contract is an MVP. The prototype should demonstrate the core

architecture for the application and provide observable value for the stakeholders. While the data model may not be finalized, a majority of issues should be resolved. We strongly recommend that security be addressed in the MVP.

3.2 Project Management Tools

To allow a systematic process of collaboration, two key development tools were used: version control and issue tracking. Git was used to store documentation and project code. Documentation was predominantly written in markdown. Development was driven by prioritized user stories. JIRA was presented at the start of the course.

3.3 Project Roles

The client is typically a person external to the course. They should have a project idea that is sufficiently interesting while being doable by a group of students that are not yet professionals. The client needs to have a clear understanding of these limitations and the potential impact on the final deliverable at the end of the second semester. Another important feature in a client is the ability to meet, either in person or via tele-conferencing, on a regular basis with the consulting team. The instructor serves as the mentor and consulting lead. The instructor employs each of the consulting teams and is responsible for overseeing the work with the goal of achieving client satisfaction for the agreed-upon budget, schedule, and scope. The consulting teams should consist of 3-5 students. Standard roles that will be filled within a team are project manager, architect, data lead, and user experience lead. Team membership and roles should change during the development process.

3.4 Project Development Artifacts

Production of artifacts is a critical part of the capstone experience. Since teams will change throughout the course, students must use artifacts developed by others as is often the case in industry. This enables a natural process of feedback as teams attempt to make use of vague or poorly executed artifacts during cross team implementation and testing. To better manage these documents, they will be created in markdown and stored along with the code base on the project repository. They will be living documents that are updated as needed as the client and development teams understanding of the project evolves.

Project Proposal: The project proposal is created based on the RFP provided by the client. This is a rapid response modeled on design contests (e.g. weekend "hack-a-thon") held by industry professionals. The expectation

is that teams will need about 6 hours to complete the work. If possible, teams should be in competition to best meet the objectives of the client.

Charter and Contract: The project charter and contract is used to document understanding of the project and guides the development process. It should clearly state the problem (what needs to be solved), and outline the key features the software solution must address.

Progress Reports: Team members will regularly document their work with billing and links to their commits.

User Stories: User stories are the primary drivers of the development effort. Each story will have an individual that is responsible for the story and its eventual implementation.

Data Diagram: The data lead is primarily responsible for designing and maintaining the documentation for the data store. Often, this will be an ER diagram for a relational data base.

Interface Sketches: The UX lead is primarily responsible for designing and maintaining user interface sketches. Typically, there will be one or more sketches for each user story.

Seed Data: Seed data is a useful but often overlooked part of project development. Using seed data allows testing of the application against a given state of the data store. The seed data can also be used to revert the data store to a default if the data store becomes damaged.

Testing: Test plans and results are required for due diligence.

4 Course Design

The goal of the redesign was to bring in aspects of designing and implementing projects based on a consulting firm delivering a project to an outside client. While it is not feasible to completely model contract development, students will get a taste of it in a more forgiving environment. The course is arranged as a series of modules guiding the process of designing and implementing a real-world project. It builds on earlier courses and is conducted over two semesters. The first semester emphasizes the design process, while the second semester culminates in the production of final deliverables. An outline of the course phases over the two semesters can be see in Fig 1. We will focus on the first semester in this paper.

The first semester will have the students establish a project charter, enter into a phased contract, develop requirements, and build architectural prototypes. Students will form into development teams with the goal of completing contracted tasks, on-time, within budget, while satisfying the client. Students will take on roles within a flexible team framework and are required to bill hours, document value, and learn to manage clients as well as the project.

The responsibility is on the instructor as mentor to find a client with a project of reasonable scope where a successful implementation is a viable result. We ask that students work hard, learn a lot, have fun, and hone many of the skills used by a software developer to make a living in applied computer science. In addition, students build pubic portfolios, get a chance to focus on and develop individual strengths, and learn to experience and appreciate the joys and challenges of working collaboratively to achieve success.

4.1 Project Timeline - First Semester

Following the consultancy model, all projects will begin with an RFP where basic requirements and motivation are set out. If clients need assistance writing one, the instructor should begin this process several months before school begins. Ideally, the client should present the RFP to the students on the first day of class. Obtaining a reasonable initial design is the first challenge to be met by the students. To start, students are broken up into small teams and develop independent designs within a very short time frame. These competing designs will be combined to create the initial design for the project. There are a couple of reasons that this approach was adopted. Producing competing designs helps to better explore the design space [9] and result in an initial design that is more likely to be viable. Students also get a brief taste of an RFP that invites multiple submissions in a phased competition.

To create the initial design, the teams will meet with the instructor and the client. The instructor will moderate as the client chooses the best parts from each of the proposals. Once the initial design is done, the features that are critical to the success of the project will be identified by the client. A minimally viable prototype (MVP) is identified and a contract is created for the next phase of the project which will implement the MVP. The instructor will play a crucial roll in the creation of the MVP. Neither the client or students are likely to be able to accurately estimate the amount of work required to complete the implementation of the MVP. It is crucial to make sure that the implementation can be completed in the requisite time. It is also likely that neither the students or clients will recognize that security and data services are high priority and should be strongly considered for inclusion in the MVP.

Implementing the MVP will be done using 3 to 4 week iterations and culminates at the end of the first semester. Teams are expected to meet regularly with their client to demonstrate progress and get client feedback. If major changes to the acceptance criteria are requested by the client, the instructor will help negotiate a modification to the contract to maintain a consistent expected level of work. As the semester winds down, students will present the MVP and the client will determine if the the acceptance criteria of the contract are met. The semester ends in a performance review addressing both

individual and team performance.

4.2 Challenges

As novices, students are typically not practiced in developing requirements and eliciting feedback from a client. As the development progresses, students may have trouble breaking tasks into manageable pieces and choosing a good implementation order. Estimating work can be a challenge even for seasoned developers. To ameliorate these challenges, scaffolding is employed providing more supports at the beginning and progressively dropping the supports as the students and teams gain experience and develop autonomy.

Architecture implementation including security is presented and addressed early in the process. Entity-relationship diagrams are developed quickly and tested early with a full set of sample data and reviewed with the client. An MVP that tests the hardest and most critical parts are implemented first, and tested in the first semester. The results of this initial deep but focused implementation and the associated lessons learned forms the basis for the detailed design requirements to be implemented in the second semester of the course. Critical features, role-based security, and some of the key services are addressed during the first semester of the course. Implementation begins very early, and the hardest parts must be tackled and solved in the first semester. This sets a firm foundation for an enjoyable, collaborative, rewarding development experience that is most likely to support student success. Solving hard parts early enables students to reach goals, supporting the intrinsic rewards of satisfying the client and perfecting useful deliverables. Progressive MVPs ensure that clients and projects can continue over a series of capstone courses, with each class of students gaining experience in proposal writing, contract development, requirement management, software implementation, and client acceptance.

5 Assessment

The course is designed to have regular assessments provided by the mentor. A students overall grade is a 50/50 mix of individual and group assessments. All assessments are done over artifacts from a public cloud-based repository for the project. A current score is available for students, but points are not given until the course nears completion.

Client involvement and feedback is critical to the success of project and ultimately the course. Clients are expected to provide formal feedback on project artifacts on a regular basis. Where possible, rubrics are provided to assist clients with providing specific, applicable, actionable feedback. During the first semester the focus is on the design documents and the implementation

of the MVP. As the basic project documents are created, the client will review them. Near the end of first semester, the client will assess the MVP, note any deficiencies, and work with the team to modify the second semester plan. At the end of the first semester, the client will review and approve the second semester project plan and, if they wish, sign a new contract for remaining work. During second semester, focus shifts to the implementation and deliverables for the application while similar reviews will occur.

5.1 Individual contributions

To measure the individual contributions, the instructor grades only documented artifacts - commits in Markdown, drawings, code, or other evidence of contributions. Contributions are associated with a single person, so teams must be careful to allow a chance for all member to commit changes that document their participation. For example, in pair programming, partners may alternate roles and commits.

Every week, students will submit detailed billing for the previous week. This will include a description with hours and links to the work product. It is expected that each student will contribute at least 15 hours a week (including 3 hours in class) broken up into no less than three sessions of at least an hour spread out over the week. Students should commit to the master branch in the shared repository after each session, so there should be at least three commits per week (more is better). Students will use a billing rate of approximately $75 per hour. (This assumes a multiplier around 2.5.) Thus, each student will generate billing of about $1100 per week. Over the full two semesters, this translates into about $150,000 for a team of five students. The mentor assesses the claims weekly and awards Individual points out of 1100, leading to a cumulative assessment of the realized dollar value over the semester. A sample billing submission has a 2-part format and is shown in Fig 2.

Links to Participating Artifacts
* Project Charter (link) * iOS app repository (link)
My top 3 person, unique contributions this week.
Commit 1 - $150 (2 hrs) - Created repo, invited team, published version (links) **Commit 2** - $375 (5 hrs) - Drafted schedule, web page gaunt chart. (links) **Commit 3** - $300 (4 hrs) - Expanded risk section of charter (links) **In Class -** $200 - On-time attendance for three class meetings.

Figure 2: Sample Individual Weekly Billing Report

5.2 Project Team Assessment

At appropriate times throughout the semester the state of the project is assessed. The assessment is user story driven with ten components. The score on a component is the percentage of user stories that satisfy the requirement.

1. The user story is assigned to a **responsible team member**.
2. **Documentation** and design reflects all client and mentor comments.
3. The client agrees with all associated **sketches**.
4. The client agrees to the **data** used. Appropriate seed data is created.
5. **Acceptance criteria** have been agreed to by the client.
6. All **implementing files** for the user story are listed.
7. All **tasks** in the project management system for the user story are done.
8. The user story is assigned to a tester to **verify** that everything works.
9. The client gives **informal acceptance** that the user story is complete.
10. The client confirms that contract **acceptance criteria have been met**.

6 Results

This effort describes a standardized approach to conducting a capstone course as a consulting effort. Scaffolding was provided early and progressively removed as students gained knowledge and experience. This provided students with an authentic experience in the context of a supportive framework. To do this, projects were conducted as consulting team competing for a contract. While many aspects of managing a project remain similar to a standard course, new artifacts were introduced. An RFP kicked off the project resulting in an initial phase contract with acceptance criteria. The contract improved interaction between students and clients, while tempering client expectations. Establishing the initial phase end goal as a minimally-viable prototype improves the students ability to decompose problems, and evaluate risk, and implement priorities. Demonstrating a prototype with observable value improved interactions with the client and motivated the students. An important aspect of providing an authentic experience was to tie their work to an estimated dollar value. This allowed us to reinforce the necessity of evenly loading all developers throughout the course. Simulated billing reinforced the expected value of their time. It motivated the use of JIRA and issue trackers to manage their time and tasks. Developing artifacts in markdown and code and using the Git distributed version control system to track changes resulted in a complete history of the evolving artifacts and participation of each team member.

7 Conclusions

Implementing the first semester of a two-part capstone computer science project as an extended consulting assignment has enabled new learning opportunities for participants. The restructured course supported continuous recognition of value and a novel approach to managing scope, schedule, budget, and risk using industry-standard tools. Future work includes the extension of this consultancy model into the second semester for final implementation and delivery.

References

[1] Lingma Lu Acheson. Student learning through hands-on industry projects. In *International Conferences on Educational Technologies 2014 and Sustainability, Technology and Education 2014*, 2014.

[2] Aaron Bloomfield, Mark Sherriff, and Kara Williams. A service learning practicum capstone. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 265–270. ACM, 2014.

[3] AT Chamillard and Kim A Braun. The software engineering capstone: structure and tradeoffs. *ACM SIGCSE Bulletin*, 34(1):227–231, 2002.

[4] Tony Clear, Michael Goldweber, Frank H Young, Paul M Leidig, and Kirk Scott. Resources for instructors of capstone courses in computing. *ACM SIGCSE Bulletin*, 33(4):93–113, 2001.

[5] Alan Dutson, Robert Todd, Spencer Magleby, and Carl Sorensen. A review of literature on teaching engineering design through project-oriented capstone courses. *Journal of Engineering Ed*, 86(1):17–28, 1997.

[6] Juliet K Hurtig and John K Estell. A common framework for diverse capstone experiences. In *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*, pages 1–6. IEEE, 2009.

[7] Larry J. McKenzie, Michael S. Trevisan, Denny C. Davis, and Steven W. Beyerlein. Capstone design courses and assessment: A national study. In *Proceedings of the 2004 American Society of Engineering Education Annual Conference and Exposition*, 2004.

[8] Joseph Mertz and Scott McElfresh. Teaching communication, leadership, and the social context of computing via a consulting course. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 77–81. ACM, 2010.

[9] Don Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2013.

[10] Norman Pestaina, Tiana Solis, and Peter J. Clarke. Assessing bs–cs student outcomes using senior project. In *2014 ASEE Annual Conference and Exposition*, pages 24.199.1–24.199.15. ASEE, 2014.

Embedding Security Concepts in Introductory Programming Courses*

Ajay Bandi, Abdelaziz Fellah, Harish Bondalapati
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{ajay,afellah,s530741}@nwmissouri.edu

Abstract

The challenges of teaching and learning introductory computer programming tend to cumulate over the years. Students should achieve a chain of ideal accomplishments including problem-solving skills, language features, structures, semantics, and coding. However, current programming practices consider security as a low-priority task and students rarely embed the predefined security requirements in their coding and software development process which may lead to insecure programs. Changing this trend in the classroom is quite challenging. As a result, a program which appears to be secure at the source code level may equate to security vulnerabilities that are often exploited by hackers. In this paper, we focus on the language-based security main features and standard security mechanisms in developing secure programs and avoiding security loopholes. This paper takes a step forward in this direction and contributes to filling the security gap.

1 Introduction

A software vulnerability is a security flaw within a software product that can be exploited by hackers and can cause cascading and recurring negative impacts in the software industry. Many software developers believe that software vulnerabilities should be addressed after the product is released, hacked, or once

errors and defects are discovered. They refer to such a policy as "release early and fix later". However, security vulnerabilities and flaws in software could happen at any point in time when the application goes live and available for the users. Several studies [15, 4] have confirmed that vulnerability disclosure may adversely affect in the long run the market performance of a software and the reputation of a firm whose software has been breached.

Developers have a misconception that security is someone else's responsibility [9]. They mostly concentrate on implementing the functionality of software systems and later releasing security updates sometimes called patches while many other users are still running old versions that might have flaws and make them prone to failure. Overall, a patch means there is one less vulnerability and access to the patch may give security advantages to an attacker. It is the responsibility of all the teams who gather requirements, design, write code, test, deploy and maintain the software [12]. Thus, the consequences of the security flaws are a spectrum of various defects that are manifested in either requirement, design, architecture, or implementation. Removing such defects and security flaws is a major challenge for software developers. One of the primary reasons that developers do not focus on security tasks is because they lack the expertise of using secure programming practices [3, 11]. Research has shown that most software [13] vulnerabilities are related to programming errors that are well understood. Such vulnerabilities could be avoided and prevented by injecting/incorporating coding standards, developing guidelines, adopting the best practices that define the programming language itself, and understanding how the language is interpreted and compiled on various runtime platforms. Moreover, research has shown [13, 16, 7] that adding security through testing is not a feasible solution and it is an incessant task for software developers. Even with automated testing tools, errors still occur in lines of code at a significant rate. Most of the current researches were focused on exploration of security by designing specific computer security courses rather than including the concepts in CS introductory programming courses.

In this paper, we present effective ways of introducing the most fundamental security-related vulnerability topics, characteristics and sources of vulnerability such as sources, severe flaws, severity and the best practices of developing secure programs in introductory computer programming courses.

Security vulnerabilities affect different types of software such as operating systems, networks, e-mail servers, software libraries, browsers, and coding, just to name a few. In the work, we focus on secure programming concepts in general and in particular on secure coding in Java. These concepts are incorporated in the introductory Java programming labs. Specifically and in terms of learning outcomes, students should be able to

- Understand common software flaws that lead to vulnerabilities.

- Identify coding mistakes and eliminate coding errors which may lead to software vulnerabilities.
- Understand how common coding mistakes can be exploited.
- Incorporate best practices and enforce a coding standard which helps programmers circumvent pitfalls and avoid vulnerabilities

From the programming perspective, the course covers the following common types of software flaws that lead to vulnerabilities: input validation errors, error handling & exceptions, memory safety violations and hashing.

2 Literature Review and Background

Some of the malicious attacks and hacking strategies are data breaches (SQL injections, XML injections), buffer overflow attacks, phishing, and sniffing, etc. Some of the security breaches negatively impacted several firms. For example, Guess.com was vulnerable to an SQL injection, allowing any user to construct a customized URL to access approximately 200,000 names and their credit card information from the website's customer database. Recently, the phishing WannaCry Ransomware attack is suspected in 150 countries by encrypting the users' hard disks impacting more than 230,000 people [3]. Heartland Payment System was attacked, and the customers' data was stolen using sniffing. The Code Red (computer worm) infected over 359,000 computers by exploiting a buffer overflow vulnerability in Microsoft Internet Information Services [17].

The primary target of the hackers and cyber-criminals is to routinely exploit the insecure code. The hacking strategies usually happen after the software is released into the market, causing a threat to the privacy of the users. When these threats are raising and discovered, organizations invest extra millions of dollars in securing their applications [8]. This also increases the cost to fix the insecure code in the existing versions as well as future releases of the software. The effects of social media and social networking sites on marketing and business have become an excellent vehicle for effective communications and have been integrated in the marketplace. Social media websites or apps allow third-party applications to collect feedback using opinion polls and track the users' activity to sponsor ads. For example, Facebook uses pixels to track the users' activity and sponsor ads. This raises a concern about users' data privacy and security. However, social media platforms are offering privacy controls, but users ignore them because of lack of knowledge and awareness. This is also applicable to the other insecure websites and apps because of lack of (adherence to) security principles. Software organizations must scrutinize the apps designed by individuals/businesses to identify any vulnerabilities or any security and policy violations. A recent example of a data firm Cambridge

Analytica which gained access to personal information of millions of Facebook users. Even though this incident is not considered to be a breach of databases but a violation of terms of use of data, privacy of the users was compromised.

3 Research Goal

Vulnerable programs are one of the leading causes of computer insecurity. Our contributions focus on developing In this research we focus on the threats to security and users' privacy in software applications that arise due to lack of secure coding practices. The best way to prevent these vulnerabilities is by improving the understanding and learning abilities of developers and users at the student level by teaching security concepts [2]. However, it is a challenge for educators to increase the number of credit hours for computer science or data science [6] degrees or to add new courses. To leverage the number of credit hours, we propose embedding secure programming concepts during the introductory programming courses. We embedded these security concepts in our lab sessions that are conducted over 2 hours weekly without slimming down or modifying the original description of the course. This helps to improve the students' ability to develop secure software from their introductory programming courses.

4 Language-Based Security Concepts

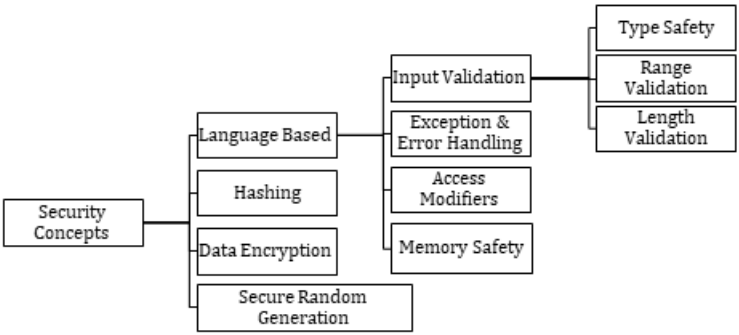


Figure 1: Classification of Security Concepts

We emphasized and embedded security concepts in the Object-Oriented Programming course (Java). We incorporated security concepts during class

time while balancing the teaching of regular material on programming concepts. Then, we embedded the implementation of security concepts in the lab sessions. The classification of security concepts are shown in Figure 1. We broadly classified the security concepts as language-based, hashing, data encryption, and secure random generation. The rationale for this classification is to cover all the basic security issues that relate to the concepts of the introductory programming course for graduate students. We further classify the language-based concepts into input validation, exception & error handling, access modifiers, and memory safety. The rationale for the classification of language-based security concepts is because students do not require any additional knowledge of these concepts as they are related to the general programming concepts. Other than the language-based security concepts, students can make use of existing Java APIs to implement in the labs. We did not concentrate those concepts in the regular class time. This gave students the similar experience of learning and using regular Java APIs. The rest of this section presents a brief description of these concepts with a sample source code that we used in labs.

4.1 Input Validation Vulnerability

This class of vulnerability is called input validation vulnerability and it occurs when a program does not check or validate user inputs. The unsafe domain of inputs may trigger a vulnerability that are commonly exploited by hackers and can arise in any programming language. The lack of validation in the input fields such as text boxes, radio buttons, checkboxes, and dropdowns, to name few are the primary reason of vulnerabilities that may lead to potential injection attacks such as SQL and XML injections. In addition to injection attacks, the invalid inputs could sometimes cause the defects in the functionality of the software [14, 5]. Typically, the class of input validation vulnerabilities includes buffer overflows, string format, and integer overflows. From the programming language perspective, the choice of the type, strong or weak, and the type checking, static or dynamic, can contribute to security problems and may affect the robustness of the software. We classified input validations into three different categories.

Strongly Typed: Strongly typed languages such as Java requires developers to infer the data type during the compilation time, i.e., developers must validate user inputs by checking the type of the data. For example:

Listing 1: Example for Strongly Typed

```
Scanner sc = new Scanner(new File(``file.txt''));
if(sc.hasNextInt()){
    //Read integer data
    int data = sc.nextInt();
```

```
}else{
    //Handle mismatch
}
```

Range Validation: Input data such as age, dates, quantity, and price need to be validated to prevent miscalculations. Range validation is not only for numeric data values but also for characters and string data types. For example,

Listing 2: Example for Range Validation

```
Scanner sc = new Scanner(System.in);
int semester;
do{
    semester = sc.nextInt();
}while(semester>=1 && semester<=4);
```

Length Validation: User inputs such as phone numbers, social security numbers, credit/debit card details are of fixed length. Validating these values are important in any software to avoid human errors. For example,

Listing 3: Example for Length Validation

```
Scanner sc = new Scanner(System.in);
long mobile = sc.nextLong();
while(new String(mobile).length() != 10){
    mobile = sc.nextLong();
}
```

4.2 Exception & Error Handling

When an error occurs such as invalid user inputs, software act in an unexpected manner in response to such an exceptional event and the code may raise an exception. Appropriate exceptions and error handling techniques are an important software component of software security. In programming languages such as C++, C# or Java, error handling is considered a form of exception handling. Let us consider an application when exceptions and errors are not handled; the complete error stack is displayed to the users. This error stack may contain all the details like which database is used or threads that reveals the implementation details. Attackers can take advantage of these error messages to create vulnerabilities [16]. For example,

Listing 4: Example for Exception & Error Handling

```
Scanner sc = new Scanner(System.in);
int semester;
try {
    semester = sc.nextInt();
} catch(InputMismatchException ex){
```

```
//User Message: Invalid semester, Enter valid semester#
    semester = sc.nextInt();
}
```

4.3 Access Modifiers

Access modifiers of classes, methods, and attributes in Java and other programming languages provide a mechanism to restrict access but may have a negative impact on the security of an application since they do not guarantee information flow control of a code. The absence of a well-defined access modifier in a Java code may break the encapsulation of that code which may lead to unexpected program behaviors. The goal of security is to prevent unauthorized access to any data. If an inappropriate access specifier is used, the attackers can change the state of the objects by accessing its variables or by calling methods. In the Java programming language, there are four access modifiers (private, protected, public, package). If global access is not required, usage of public access modifier to the state variables of a class should be avoided as it allows any other object to change its state. Alternatively, it is better to use public setter methods to mutate the state of instance variables by performing any required validations.

4.4 Buffer/Integer Overflow

Java language is considered to be memory safe regarding array bounds and pointer dereferences. However, it is required to explicitly handle integer overflows while calculating and assigning values that involve large numerals. A buffer overflow occurs when the user is trying to store data into the buffer than its limit. It is one of the most common vulnerabilities. In every programming language, a datatype is pre-defined to hold certain bytes (for instance, integer type in Java, can hold 4 bytes with any value in the range of -2,147,483,648 to 2,147,483,647. When assigning a value to a variable, Java would throw a syntax error during the compile time if the value is out of the range [7, 17].

4.5 Hashing

Confidential data like passwords and digital signatures are used to verify the authenticity of an account or a document. If these details are stored as clear text in the database, a hacker could hack the passwords during the data breach. Instead, use a hash function to generate a hash code and store in the database. Later, this hash code is compared with the new hash code generated from the data provided by the user. Java API provides MD5, SHA-1, SHA-256

algorithms to create the hash values. An example source code snippet for SHA-1 is shown below.

Listing 5: Example for Hashing

```
MessageDigest md = MessageDigest.getInstance("SHA");
byte[] digest = md.digest(password.getBytes());
String encryptedPassword = new String(digest);
```

4.6 Data Encryption

Encryption can secure the transmission of data through a network. Java Cryptography Architecture (JCA) [11] allows implementing several prominent symmetric and asymmetric algorithms for verification of messages and encryption/decryption of data. Java API includes encryption algorithms like AES, Blowfish, DES, RSA, DESede, etc. Sample source code for data encryption using AES algorithm is provided below.

Listing 6: Example for Encryption

```
KeyGenerator keygen = KeyGenerator.getInstance("AES");
SecretKey aesKey = keygen.generateKey();
Cipher aesCipher;
// Create the cipher
aesCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
// Initialize the cipher for encryption
aesCipher.init(Cipher.ENCRYPT_MODE, aesKey);
// Our cleartextbyte[] cleartext =
    "This is just an example".getBytes();
// Encrypt the cleartext
byte[] ciphertext = aesCipher.doFinal(cleartext);
```

4.7 Secure Random Number Generation

Pseudo random number algorithms are used to generate a sequence of random values. While the sequence generated by these algorithms may appear to be random, at their core they are not. Having an insecure or deterministic random generator cause vulnerabilities in applications. Java API provides a SecureRandom class to generate a secure pseudo random number generator [11]. A sample code snippet is shown below.

Listing 7: Example for Secure Random

```
SecureRandom random = new SecureRandom();
 byte bytes[] = new byte[20];
 random.nextBytes(bytes);
```

5 Case Studies

The methodology of the case studies is adopted from [10]. In this research, we proposed emphasizing security concepts to students and embedding security-related scenarios in lab assignments. We have conducted our case studies in a graduate course of a total of 90 students, where students are taking Object-Oriented Programming course (Java emphasis). Approximately 30% of the graduate students who enrolled in this course are computer science undergraduate majors and the remaining 70% are non-computer science undergraduate majors. The steps involved were as listed below:

1. Students are pretested for their current level of understanding security concepts using survey [1].
2. Emphasize on security concepts to the students was done while teaching the course while balancing the regular programming concepts.
3. Students are given a total of 12 lab assignments embedding scenarios related to security concepts along with regular programming language concepts. Different kinds of input validation scenarios are included in the Control Structures, Interfaces & Abstract Classes, and Sorting & Equals labs. Exception & Error handling scenarios are included in the Casting & Exceptions and Sorting & Equals labs. Hashing and Random Generators are included in the Interfaces & Abstract Classes. Buffer Overflow scenario are included with Recursion tracing problems.
4. Students are post-tested for their level of understanding security concepts using survey [1].

6 Results and Analysis

After collecting the data from our case studies, we presented our results in a comprehensive tabular format shown in Tables I and II.

Level of Understanding	1 (Never heard of)	2 (Exposure)	3 (Familiarity)	4 (Knowledgeable)	5 (Mastering)
Pretest	23%	36%	22%	16%	4%
Post-test	0%	1%	7%	42%	51%
Difference	-23%	-34%	-15%	26%	47%

Table I: Students' Level of Understanding of Security Concepts

Table I shows the results for the overall level of understanding of all the security concepts. The pretest results show that around 59 students out of 73 were below knowledgeable. Later, we emphasized the security concepts in the class, embedded those concepts in labs and asked students to solve the

lab exercises. After the post-test, all the students have substantially strengthened their technical knowledge of security concepts mentioned in this paper. Both students pretest and post-test results were proved by the instructor. The "Mastering" level did improve by 47%. These students in level 5 moved from other levels. The number of students who are knowledgeable about the security concepts is also increased by 26% and these students moved from lower levels. The increased percentage in green color shows the substantial positive increase of understanding level. The red color graphs are considered as positive decrease highlighting the number of students moved from the lower three categories (never heard of, exposure, familiarity) to the knowledgeable and mastering categories. Overall there is no negative impact of introducing security concepts in the introductory programming class for graduate students.

Table II shows the comprehensive pretest and post-test results of individual security concepts. The "Mastering" level of students did improve in all the individual security concepts. From the results, most of the students' understanding level increase in the input validation, category.

Level of Understanding		1 (Never heard of)	2 (Exposure)	3 (Familiarity)	4 (Knowledgeable)	5 (Mastering)
Input Validation	Pretest	1%	21%	30%	29%	19%
	Post-test	0%	1%	6%	32%	62%
	Difference	-1%	-19%	-25%	3%	42%
Exception & Error	Pretest	10%	16%	33%	34%	7%
	Post-test	0%	1%	8%	48%	43%
	Difference	-10%	-15%	-25%	14%	36%
Access Modifiers	Pretest	7%	14%	29%	37%	14%
	Post-test	1%	0%	6%	38%	55%
	Difference	-6%	-14%	-23%	1%	41%
Hashing/ Data Encryption	Pretest	14%	21%	33%	23%	10%
	Post-test	0%	1%	8%	44%	47%
	Difference	-14%	-19%	-25%	21%	37%
Random Generators	Pretest	8%	29%	25%	22%	16%
	Post-test	0%	1%	7%	48%	44%
	Difference	-8%	-27%	-18%	26%	27%
Buffer Overflow	Pretest	12%	22%	33%	22%	11%
	Post-test	1%	4%	18%	38%	38%
	Difference	-11%	-18%	-15%	17%	27%

Table II: Students' Level of Understanding of Individual Concepts

In all the security concepts, there is a definite increase in the understanding levels of students in mastering and knowledgeable categories. The red color graph in each category shows that the positive decrease highlighting the students in those levels move students to upper levels. In our study, none of the

students move from higher level to lower levels. Therefore, there is no declining performance of any student.

7 Conclusion

The proposed approach of embedding security concepts in introductory programming courses and the practical implementation of those concepts in labs for graduate students has provided evidence in strengthening the students understanding and learning considering the security concepts without compromising the programming skills. The empirical evidence shows that students' understanding and learning improved from lower levels to higher levels. Students are also comfortable in handling security issues such as input validations, hashing, exceptions, usage of access modifiers, and generating secure random numbers. Our study also highlights providing insights on how to teach security concepts without increasing the number of credit hours in the computer science curriculum. As future research, we propose to identify any shortcomings and pitfalls in this approach by extending this study to larger computer science undergraduate majors for a more extended period.

References

[1] Pre and post survey. `https://goo.gl/forms/rBcJuHWDSt21hvO12`.

[2] Java cryptography architecture, 1993,2018. `https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html`.

[3] *OWASP Secure Coding Practices Quick Reference Guide*. 2010.

[4] Ashish Arora, Ramayya Krishnan, Rahul Telang, and Yubao Yang. An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure. *Information Systems Research*, 21(1):115–132, 2010.

[5] C. Warren Axelrod. *Engineering Safe and Secure Software Systems*. Artech House, 2012.

[6] Ajay Bandi and Abdelaziz Fellah. Crafting a data visualization course for the tech industry. *Journal of Computing Science in Colleges*, 33(2):46–56, 2017.

[7] Jon Brodkin. The top 10 reasons web sites get hacked. *Network World*, 24(39):1–20, 2007.

[8] Brian Chess and Jacob West. *Secure Programming with Static Analysis*. Pearson Education, 2007.

[9] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. pages 119–129. DARPA Information Survivability, 2000.

[10] Abdelaziz Fellah and Ajay Bandi. The essence of recursion: Reduction, delegation, and visualization. *Journal of Computing Science in Colleges*, 33(5):115–123, 2018.

[11] Katerina Goseva-Popstojanova and Andrei Perhinschi. On the capability of static code analysis to detect security vulnerabilities. *Information and Software Technology*, 68(C):18–33, 2015.

[12] Samanvay Gupta. Buffer overflow attack. *IOSR Journal of Computer Engineering*, 1(1), 2012.

[13] Elisa Heymann, Barton P Miller, and Jim Kupsch. 10 common programming mistakes that make you vulnerable to attack. *Condor Week*, 2012.

[14] Clifton Phua. Protecting organisations from personal data breaches. *Computer Fraud & Security*, 2009(1):13–18, 2009.

[15] Rahul Telang and Sunil Wattal. An empirical analysis of the impact of software vulnerability announcements on firm stock price. *IEEE Transactions on Software Engineering*, 33(8):544–557, 2017.

[16] Jason Earl Thomas. Individual cyber security: Empowering employees to resist spear phishing to prevent identity theft and ransomware attacks. *International Journal of Business and Management*, 13(6), 2018.

[17] Kenneth A Williams, Xiaohong Yuan, Huiming Yu, and Kelvin Bryant. Teaching secure coding for beginning programmers. *Journal of Computing Sciences in Colleges*, 29(4):91–99, 2014.

Introducing Fundamental Computer Science Concepts Through Game Design*

Fei Cao[1], Dabin Ding[1], Michelle Zhu[2]
[1]School of Computer Science and Mathematics
University of Central Missouri
Warrensburg, MO 64093
{fcao,dding}@ucmo.edu
[2]Computer Science Department
Montclair State University
Montclair, NJ 07043
zhumi@montclair.edu

Abstract

Understanding fundamental Computer Science (CS) concepts and nurturing computational thinking skills are essential to learning to solve various science and engineering problems. However, many computer science curricula, especially in the early college years, do not provide students with an engaging and fun learning environment especially for traditionally underrepresented minorities in CS. Meanwhile, educational games have become an important teaching tool in a wide variety of disciplines. We plan to integrate CS concepts and skills such as debugging, data structure, and parallel and distributed computing concepts into our game design courses. Our experiences indicate that (1) students tirelessly strive to understand the CS concepts in order to design game challenges and (2) the virtual reality (VR) immersion experience is enormously engaging and fun.

1 Introduction and Motivation

Computer Science is a fast-growing discipline and has been penetrating into and interacting with almost every aspect of our daily life. However, we found that very few early courses provide our students with opportunities to apply innovative thinking to problem-solving in an attractive and fun way. The potential of video games make them a unique and engaging CS teaching platform as they can bear student learning objectives under multiple levels with increasing difficulties, allow repeated practices for proficiency, adapt to personal learning pace and background as well as keep track of the progress with the strength and weakness for each student[7, 9, 1]. [11] studied the 12 factors contributing to success in introductory computer science courses and suggested game playing was predictive of success. Therefore, we plan to enhance our current game courses to help students learn more CS concepts at both undergraduate and graduate levels. Our preliminary work has been included in a two-page poster paper[2].

2 Game Course Description

The entire class is assigned to design and build a role-play adventure educational game using game development tools including Blender [5] and Unity [10]. Students work in teams. There are several phases from design to model and to prototype, namely game concept, CS topics, story and characters, gameplay, level design, and interface design as shown in Figure 1. Pre-selected CS topics such as programming, data structures, and Parallel and Distributed Computing (PDC) concepts will be covered. Collaboration among different teams is critical to ensure the consistency for level advancement. A world map is designed with multiple levels on each continent. Each team is responsible for the development of one level of the game. We present two example student projects here as case studies.

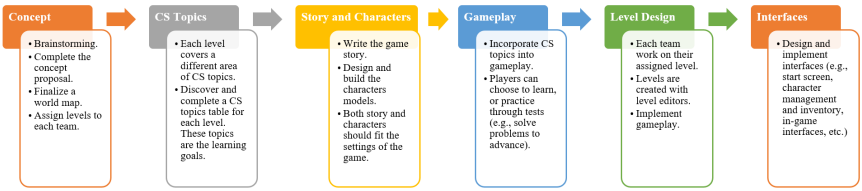
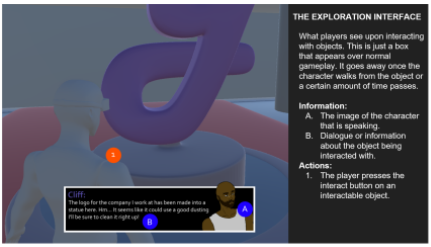


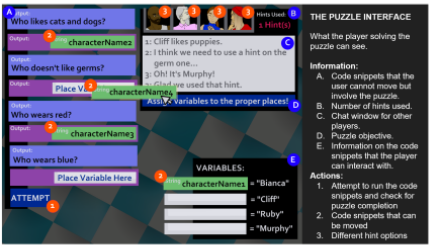
Figure 1: CS Educational Game Design Phases

2.1 Case Study I

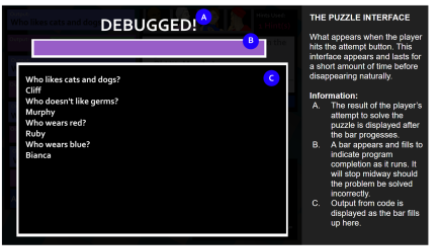
Programming concept of debugging: The first example student project focuses on programming topics. It was designed to teach the players how to problem-solve in the world of programming. The game is set in an augmented reality of the Silicon Valley area. A team of players wear VR headsets and play a "simulation" of their world. The team ends up getting stuck when the VR headsets break due to an unexpected glitch which may cause various bugs and errors in the system's code. Each level in the game takes place in a different semi-open world location, with the difficulty of puzzles scaled to each environment. The design of this game focuses on teaching the concepts of error-reading and checking through debugging the code tied to various in-game objects. These concepts are presented primarily via puzzles. The players navigate the in-game world in search of escapes. When they find out the cause of the game's bugs, they must move toward the enemy headquarters, where the glitch that is trapping them within the game resides. The players are given a programming puzzle when interacting with some objects (as shown in Figure 2a), as well as after a combat segment. The team must pass in order to "debug" or "fix" the object or enemy so that it is functional and docile again. The puzzle elements are set up in a Scratch-like interface, in which code is viewed via visual code segments. Objects related to the puzzle may be moved and changed, and an error code acts as a hint. One player is chosen to solve the puzzle. Other team members may send messages to this player to help or cheer him on. The player chosen to solve the puzzle is determined at random, though a player who has not solved



(a) Player Interacts with A Puzzle Point



(b) Four Players Collaborate to Solve Programming Puzzle



(c) Players Successfully Solved the Programming Puzzle

Figure 2: A Student Game Challenge Example on Debugging

a puzzle in a while will be set to a higher priority level to be chosen. Once chosen, this player's priority level will be reset. This is to ensure that each player will get a chance to contribute and that the game is not completed by one single player. A sample puzzle solving interface is shown in Figure 2b where four players collaborate to debug a program, and Figure 2c shows the interface those players see once they successfully solve the puzzle.

2.2 Case Study II

Parallel and Distributed Computing concept of workload balance: A second example student project focuses on PDC topics. It was designed to illustrate the challenge of navigation and searching for robots. The scenario of the game is that all players are trapped in a virtual world in VR and cannot get out. The players need to navigate the world, learn the key PDC concepts, and use the acquired knowledge to grab resources, solve puzzles, and overcome obstacles to escape traps. We experimented the challenge of applying static load balancing concepts with the same set of students in the game course with no prior knowledge of PDC. Students were asked to first act as the players and then as the designers.

As players, students were asked to efficiently distribute ten tasks to three robots. Each task has a randomly generated computation time. The three robots have identical computation power and execute assigned tasks in parallel. Students were also required to record their scheduling time, minimum execution time, and the strategies they used. We designed three missions for experimentation: 1) schedule a batch of ten tasks simultaneously; 2) schedule the ten tasks in sequential order, where there are no dependencies among the tasks; and 3) schedule the ten tasks with dependencies according to a Directed Acyclic Graph (DAG).

Due to uneven workload distribution of specific solutions, it is possible that some robots will complete their tasks before others and become idle. Since the total time needed will be determined by the slowest robot, the objective is to have all three robots finish the assigned tasks at approximately the same time with an equal workload. Based on student feedback, most students said that the leading strategy they used was to balance the workload of the three robots to get the minimum execution time. They learned problem-solving and how to efficiently schedule the tasks. Among the 30 students who took the challenge, most students said that they used similar, yet slightly different strategies for mission 1 and 2; mission 3 is more complicated and took them the longest time to find an optimized schedule. Most of them scheduled the tasks layer by layer, with adjustments for each layer. They also mentioned that they may not be able to find an optimized solution for mission 3 if the number of tasks scaled up.

Students were then asked to re-design the challenge after they acquired the knowledge for solving the above three missions. They came up with all kinds of creative gameplay to apply the concept into a scenario. They also took the varying backgrounds of players into consideration and designed different difficulty levels. Through the games they re-designed, players should learn the concept at their own pace and learn how to apply the acquired knowledge and thinking skills in a problem-solving scenario. Players are put under pressure and engaged via challenges, stories, and entertainment. The re-designed games also provide incentives for players to go up to higher difficulty settings. That gives players a chance to identify patterns or formulas by trying different techniques in their placements of the process blocks.

(a) Robot Navigation and Searching Gameplay

(b) Apply Static Load Balancing Concept to Distribute Tasks Among Three Robots

Figure 3: A Student Game Challenge Example on Load Balancing

Figure 3 shows the game design of the above challenge example of applying static load balancing concepts among multiple processors in order to remove obstacles and access the next area. The purple items in Figure 3b are computation tasks of different sizes and computation time. Each player will strategically assign each task to a robot so that the summation of the total computing time for each robot will be the same or close. The game will compute the total time for each player and determine the winner. Given nine tasks ordered with the following computing time (32, 30, 20, 14, 11, 8, 5, 3, 2), for instance, a good systematic distribution would be to assign the tasks in the descending order for each robot and then reverse the order of the robots for the next round of task distribution until all tasks are gone. Consequently, we will get robot A with tasks (32, 8, 5), robot B (30, 11, 3) and robot C (20, 14, 2). The total time for each robot will be 45, 44, and 36 respectively. The final task completion time for the three robots will be the one with the longest time which is 45 for robot A. If several players

complete simultaneously, the one with the shortest time distributing the tasks wins and gets more resources than others.

3 Experience and Future Work

We observed that most students found the courses exciting. They needed to study the CS topics thoroughly in order to design the challenges. More students, including non-CS majors enrolled in the courses. Students not only learned game design and development but also enhanced their programming skills and learned CS concepts. Students were able to comprehend the concepts more easily and reflect on their mistakes for better performance in the next rounds. For future work, we would like to design more challenges covering additional key CS skills and concepts. We are also aware that certain types of problems work better under team competition mode, while others require deep individual reflexive thinking. Based on that, we would like to design various game modes and difficulty levels to allow students to play and learn at their own pace. As women are underrepresented in computer-related programs [4, 3], it is also our interest to include more game themes to attract more female students to participate [6, 8].

4 Acknowledgement

We gratefully acknowledge the support from the Early Adopter for Multi-Course and Multi-Semester PDC award by NSF/TCPP program.

References

[1] Ashok R Basawapatna, Kyu Han Koh, and Alexander Repenning. Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 224–228. ACM, 2010.

[2] Fei Cao, Dabin Ding, and Michelle Zhu. Engaging students in parallel and distributed computing learning by games design using unity. *Poster paper at EduHPC-18: Workshop on Education for High-Performance Computing*, Dallas, TX, Nov. 2018.

[3] Maria Charles and Karen Bradley. *A matter of degrees: Female underrepresentation in computer science programs cross-nationally.* na, 2006.

[4] J McGrath Cohoon. Toward improving female retention in the computer science major. *Communications of the ACM*, 44(5):108–114, 2001.

[5] Blender Foundation. Home of the blender project - free and open 3d creation software. `https://www.blender.org/`.

[6] Tracy Fullerton, Janine Fron, Celia Pearce, Jacki Morie, et al. 11 getting girls into the game: Toward a "virtuous cycle". 2008.

[7] Mark Overmars. Teaching computer science through game design. *Computer*, 37(4):81–83, 2004.

[8] Marina Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1):1–12, 2009.

[9] Kathryn T Stolee and Teale Fristoe. Expressing computer science concepts through kodu game lab. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 99–104. ACM, 2011.

[10] Unity. `https://unity3d.com/`.

[11] Brenda Cantwell Wilson and Sharon Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. In *ACM SIGCSE Bulletin*, volume 33, pages 184–188. ACM, 2001.

How Learning Works: Applying Cognitive Psychology Theory to Computer Science Course Structure*

Conference Workshop

Jennifer McKanry, Gretchen Haskell, Dasha Kochuk
Center for Teaching and Learning
University of Missouri – St. Louis
St. Louis, MO 63121
{mckanryj, haskellg, kochukd}@umsl.edu

Many traditional teaching strategies, such as lecture, can be made more effective by incorporating active learning strategies derived from cognitive and educational psychology research. Understanding the basics of how learning works can help restructure your class to take advantage of your students' strengths. Additionally, you can build in homework exercises that have been shown to be far more effective in helping students learn than rereading and highlighting. In the following paragraphs a few of these concepts will be described.

The human attention span in a lecture environment deteriorates after approximately ten minutes. By incorporating active learning and movement into a class at regular intervals you can effectively reset this attention clock. You can also stimulate neuropathways through the idea of retrieval practice (testing effect). This involves asking your students, in low-stakes ways, to recall information without resources. Even if they are incorrect, this will build stronger pathways to the information that have been shown to develop longer term retention. Examples might include quizzing or writing of a one-minute paper where students recite everything they remember on a topic. This will also help them space out their studying (distributed learning) which is a more effective learning strategy than cramming. Problem based learning is also a very effective way to promote deeper learning by students. This involves giving students a unique problem to solve prior to explaining to them how they should solve it. This struggle helps students have more motivation to solve the problem and

understand the logic behind the solution. Struggle, while uncomfortable for students, is a very effective learning tool and makes learning more "sticky".

Finally, the strategy of introducing application examples before theory. While traditional lecture introduces theory before showing examples of application, this is actual not the best way for students to learn. Students do not know how to store or encode theory presented out of context, therefore, they remember far less. However, when an application example is presented first, students then have a context in which to place the theory when reviewed. This short list is only a brief example of the simple steps that can be taken to adjust courses to improve student outcomes.

References

[1] Susan A. Ambrose, Michael W. Bridges, Michele DiPietro, Marsha C. Lovett, and Marie K. Norman. *How learning works: Seven Research-based Principles for Smart Teaching.* Jossey-Bass, San Francisco, CA, 2010.

[2] Peter C. Brown, Henry L. Roediger III, and Mark A. McDaniel. *Make It Stick: The Science of Successful Learning.* The Belknap Press of Harvard University Press, Cambridge, MA, 2014.

[3] Joshua R. Eyler. *How Humans Learn: The Science and Stories behind Effective College Teaching.* West Virginia University Press, Morgantown, WV, 2018.

CyberReady StL Curriculum: Tutorial, Best Practices, and Results from Initial Deployment*

Conference Workshop

Rebecca Dohrman[1], Paul Gross[2], Steve Coxon[3], Chris Sellers[4], Christi DeMuri[5], Robyn Ray[4], Dustin Nadler[1]
[1]College of Arts and Sciences, Maryville University
[2]Gross Code and Education
[3]School of Education, Maryville University
[4]Jennings High School
[5]Ritenour High school
St. Louis, MO 63144
{rldohrman,scoxon,dnadler}@maryville.edu

This workshop will walk participants through the development and recent deployment of the CyberReady StL curriculum which is built on the Raspberry Pi platform to introduce students to the basics of coding in Python, the Raspberry Pi platform (with SenseHat), and networking in order to help students be more cyberready and to prepare them for subsequent computing curricula (i.e. CyberPatriot). The tutorial will be presented by a team of researchers from Maryville University, a computing expert who was on the development team for the curriculum, and three educators who deployed the curriculum in the Fall 2018. The team will talk through results from the pre- and post- tests about attitudes related to computing as well as cyberreadiness skill.

Introduction to Cloud-Based Machine Learning*

Conference Workshop

Saty Raghavachary, Jeffrey Miller
Computer Science Department
University of Southern California
Los Angeles, CA 90089
{saty, jeffrey.miller}@usc.edu

Machine Learning (ML) is transforming society in fundamental ways. From recommendation engines to self-driving cars to e-commerce to voice-driven personal assistants, they are seemingly ubiquitous. There is a huge demand, from students across multiple disciplines (including non-CS, non-technical ones, such as business, music, health, law), for courses that introduce them to the fundamentals of ML. This tutorial aims to get participants started on ML, using a mix of lecture and hands-on examples.

In this tutorial, we will present the basics of machine learning, focusing on neural networks, and walk the attendees through two complete machine learning examples: a basic one that uses just three binary inputs to train a small network coded from scratch, and another one that uses the expressive Keras library to create and train a model to learn to identify cats and dogs from a dataset of input images, and use the results to have the model classify new images.

The tutorial should be of interest to colleagues who are considering developing a beginner ML course, or incorporating small ML projects into their existing courses. We will use Google's 'Colab' for our two examples - Colab is a powerful 'cloud' engine that is GPU-enabled for accelerated computing, and is available via just a standard GMail/GDrive account. Attendees will learn the fundamentals of machine learning, specifically the use of neural networks for training and classification; use of Colab via a standard web browser, to launch ML projects on the Google cloud; use of 'Jupyter' notebooks ('computable documents') to house code as well as related text and images.

The tutorial is expected to last 60-90 minutes.

SASS (Syntactically Awesome Style Sheets)*

Conference Tutorial

Jane O'Donnell
Department of Computer Science
Multimedia and Web Design Program
St. Charles Community College
Cottleville, MO 63376
jodonnell@stchas.edu

SASS is an easy-to-use open source styling language that helps reduce the repetition and maintainability challenges of traditional CSS. SASS allows you to scale styles when working on big web development projects, making it faster and more efficient to write reusable styles from scratch for smaller projects. SASS is a preprocessor scripting language that compiles into Cascading Style Sheets (CSS) using a GUI compiler app many of which are open source. Like object-oriented languages, SASS uses variables, nesting, mixins and functions to write reusable styles. Once completed the SASS file is then translated into a CSS file using a compiler.

Participants in this tutorial will be presented with the basic syntax of SASS using a simple web editor Notepad++ with an introduction of a variety of SASS open source compilers including Koala. When the SASS Script is interpreted by the compiler, the resulting CSS will be applied to a web page.

References

[1] Hampton Catlin, Natalie Weizenbaum, and Chris Eppstein. SASS. `https://sass-lang.com`. Retrieved January 17, 2019.

[2] Ethan Lai. Koala. http://koala-app.com/. Retrieved January 17, 2019.

[3] Taisiya. 10+ best tools and resources to compile manage SASS, LESS, and Stylus – CSS preprocessors. https://graygrids.com/best-tools-resources-compile-manage-sass-less-stylus-css-preprocessors/. Retrieved January 17, 2019.

Easy Handwriting Recognition*

Nifty Assignment

Eric D. Manley, Timothy Urness
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
{*eric.manley,timothy.urness*}*@drake.edu*

Abstract

In this assignment, students create a GUI which allows the user to draw handwritten digits using a mouse and then attempts to recognize the character using a CS1-accessible machine learning algorithm. The assignment can be used to emphasize file input, 2D lists/arrays, and/or GUIs. The assignment is nifty because it uses real data with a simple algorithm to achieve compelling results for a familiar application.

1 The GUI

The students create a GUI consisting of a canvas which responds to click-move mouse events by changing the color of the canvas pixel and capturing the corresponding entry in a 2D-List of 0s and 1s, a button which runs the prediction algorithm, and a label to display the algorithm's guess. The example GUI in Figure 1 was made using the Python *tkinter* module, though any GUI with a similar canvas (and any programming language) will work.

2 The Data

The user's drawing is compared against a set of handwritten digit samples from the well-known MNIST database [1], which have been reformatted for CS1-accessible reading. We converted the data from grayscale to black and white for easier comparison, reduced the number of samples (from 60000 to 2000), and converted it from binary to a comma separated values (csv) file

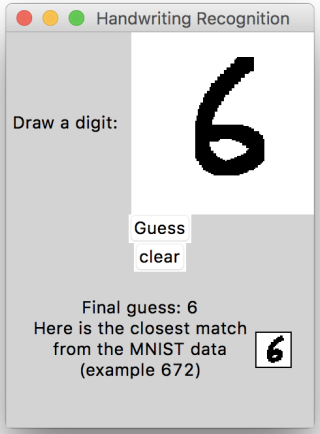


Figure 1: The GUI and Data

where every 29 lines consists of a line with the digit represented and then 28 lines representing the rows in the 28x28 pixel image (see Figure 1). The resulting data set has an easy-to-understand format, where the handwritten characters can be seen visually in the 0s and 1s, and students see immediately how it should be read in as 2D-lists.

3 The Prediction Algorithm

Students write prediction code as follows:

- Create a function computes a similarity score between the drawing 2D list and a sample from the data set. You can simply count the number of pixels that agree or weight some more heavily than others (e.g., 5 points for black pixel agreement, 1 point for white). As an optional extension, students can be given the freedom to experiment and define similarity in any way they choose.
- Loop through each of the samples in the data set, compute the similarity between that sample and the drawing, and keep track of the sample with the best similarity score. The digit corresponding to the best similarity score is then displayed as the prediction.

This algorithm is a variant of the K-nearest-neighbor (KNN) machine learning algorithm with $k = 1$ and a custom similarity function. Other variations of KNN are known to perform very well on this data set [1].

References

[1] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database. `http://yann.lecun.com/exdb/mnist/`.

Streaming TV Services*

Nifty Assignment

Kendall Bingham
School of Computing and Engineering
University of Missouri - Kansas City
Kansas City, MO 64110
binghamkl@umkc.edu

Abstract

Cutting the cord has become very popular in the last few years. With so many choices streaming TV services with multiple packages how does a consumer know which is the best value for their viewing habits? This assignment allows students to create a program to find the best streaming service for a user.

1 Student Skills

This was a CS101 program given about 3/4 of the way through the semester in Python:

- Reading Files, in this example .csv files were used
- Dictionaries

2 Input Files

There were 4 types of files given for their project. Some types consisting of multiple files. For instance ProviderName.csv has a csv file for each provider, so there could be dozens of this type of file.

1. Provider.csv: contains all the providers, prices, and the name of the csv file that contains the channels that provider carries.

2. {ProviderName}.csv: one for each provider name, simply has all the channels for that streaming service.

3. shows_channel.csv: list of TV shows with the channel that it airs new episodes on. We don't cover re-runs.

4. {username}.csv: multiple user files are provided to test with. These contain shows they like to watch with an associated weighting. The higher the weighting the more important the show is to the user.

3 The Output

This assignment had 2 parts; Cost Per Channel, and User Weighted Choice.

3.1 Cost Per Channel

For this output they only needed to read the provider file for name, price, and quantity of channel to calculate the cost per channel, sorted from low to high. Figure 1 shows the Cost Per Channel output.

Cost per Channel Summary

Service	CPC	Channels	TTL Price
Sling Orange	0.4762	42	20.00
Sling Blue	0.4902	51	25.00
Direct TV Now Live a little	0.5303	66	35.00
Direct TV Now Go Big	0.5455	110	60.00
Direct TV Now Just Right	0.5814	86	50.00
Direct TV Now Gotta Have It	0.6034	116	70.00
PS Vue Elite	0.6548	84	55.00
Hulu with Live TV	0.7143	56	40.00
PS Vue Core	0.7627	59	45.00
YouTube TV	0.7955	44	35.00
PS Vue Access	0.8511	47	40.00
PS Vue Ultra	0.8824	85	75.00

Figure 1: Cost Per Channel

3.2 User Weighted Choice

The second part of the assignment was somewhat more challenging. The program asked for an input file of a user's favorite show and a weighting. The higher the weight, the more important the show was to that person. Each streaming service score would be the sum all the weights of shows that they carried.

For instance, if we have 2 Providers; Sling and DirectTV. The user likes a couple of shows that we'll call Apples and Bananas. Apples is something they like to watch, but don't need, so they weighted it a 1. Bananas is as show they cannot miss, so they weighted it 5. Sling has Apples, but not Bananas, DirectTV has Apples and Bananas.

$$SLING = 1$$

$$DIRECT\_TV = 6 = 1 + 5$$

The output is then sorted by the cost per point as shown in Figure 2.

User Cost per Point Summary

Service	CPP	Points	TTL Price
Sling Blue	3.1250	8	25.00
Direct TV Now Live a little	4.3750	8	35.00
Direct TV Now Just Right	6.2500	8	50.00
Sling Orange	6.6667	3	20.00
YouTube TV	7.0000	5	35.00
Direct TV Now Go Big	7.5000	8	60.00
Hulu with Live TV	8.0000	5	40.00
PS Vue Access	8.0000	5	40.00
Direct TV Now Gotta Have It	8.7500	8	70.00
PS Vue Core	9.0000	5	45.00
PS Vue Elite	11.0000	5	55.00
PS Vue Ultra	15.0000	5	75.00

Figure 2: Cost Per Point

4 Conclusion

Most students found it to be a fun and relatable assignment. The difficulty of the assignment is easily modified as well. If the data structures are too complex, then combining some of the information into 1 or 2 files would work. The algorithm and methods can easily be tweaked to fit other languages or lessons. The large number of records can be intimidating to students, it would be advised to include a smaller subset for students to get a familiar with.

4.1 Student Feedback

The assignment was very well received by the students 58%, rating it as one they liked. These were some of the more interesting student quotes.

> Awesome introduction to data structures. -Anon

I like this program .it was my favorite -Anon

I felt like this program was a little to easy... -Anon

very difficult -Anon

The original assignment and files can be found at the following repository: `https://github.com/kbingham-umkc/StreamingTVService`

Applying Asymmetric Encryption Algorithm Using Kryptos*

Nifty Assignment

Imad Al Saeed
Computer Science Department
Saint Xavier University
Chicago, IL 60655
773 298-3393
alsaeed@sxu.edu

Abstract

This assignment is for a course on Cybersecurity, for students with no prior knowledge with Cryptography (most interested in assignments for CS1-CS2). The course instructor explains the Asymmetric Encryption Cryptography concepts as solutions for the key distribution to the students. Also, the course instructor explains how the Public Key Cryptography gives another way to communicate privately without having to share secret information prior to the communication. The instructor provides the students with the following link to download the Kryptos software `https://tinyurl.com/yxdr997w`. Students should download Kryptos version (2.0) (Figure 1) as the newer version may not work on their systems.

1 General Scenario

Students should pair with a classmate and send each other encrypted messages using a program called Kryptos. Both students should follow the same steps and use the same algorithms and parameters to generate their keys (Public and Private Keys).

One student should be a sender and the second one should be a receiver. Both students should use RSA (Rivest–Shamir–Adleman) as one of the first

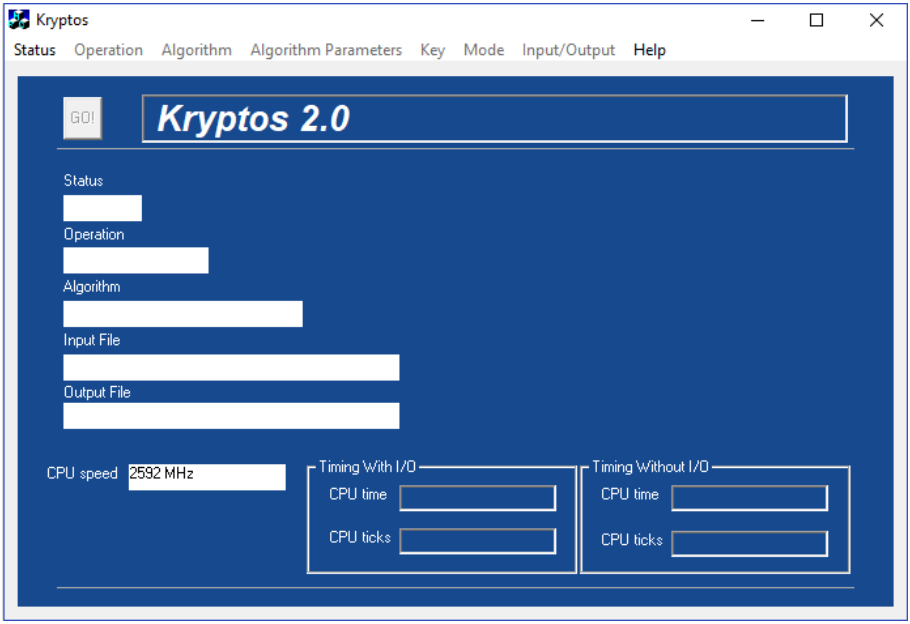


Figure 1: Kryptos Software

public-key cryptosystems used for secure data transmission and SHA-1 (Secure Hash Algorithm 1) algorithm to generate their public and private keys. Students have to name their key file as Lastname_Key and save it under their working folder. Students should send the public key file to each other via email. Upon receiving the public key via email, each student can send encrypted messages to each other over a public channel (the Internet). Each student should load his/her partner's encrypted message file and the public keys file to Kryptos software.

2 Assignment Setup

Part 1: Generating Public and Private keys

First, students should create a working folder and name it (Asymmetric_Encryption). Second, students should start a Kryptos program and set the Status parameter to "Sender", Operation parameter to "Public key encrypt", Algorithm parameter to "RSA", "No padding", and "SHA-1", and the Algorithm parameter should be left as is (using the defaults setting). Regarding to the Key parameter, students should select "Set Key", click on the "Generate

Key" from the pop-up menu, and then click on "Save key". Students should name the key files as Lastname_Key and save it under the Asymmetric_Encryption folder. The program will generate two keys (Public key and Private key) named as Lastname_Key.private and Lastname_Key.public located under the working folder. Finally, each student should send only a public key file (Lastname_Key.public) to each other via email.

Part2: Message Encryption Process

Each student should use their partners public key that he/she received via email to encrypt his/her message as follows: First, each student should use the Notepad program to create a text file. Each student should write a message in that text file and save it as Lastname_Plain.txt. Second, each student should repeat the exact same steps mentioned in step 1 including the parameter and algorithms until Generating Key step. At the Greater Key, students should load his/her plain text file and the public key file that he/she received from his/her partner and select "input file" to load the plain text message file and select "New" for the output File and name it as Lastname_Cipher.enc. Finally, students should hit GO. The new encrypted message file will be saved inside the working folder. Each student should send his/her encrypted file (Lastname_Cipher.enc) to his/her partner.

Part3: Message Decryption Process

After the partners receive the encrypted message file from each other, he/she should use his/her own private key to decrypt the encrypted message using the Kryptos software by setting the Status parameter to "Receiver", Operation parameter to "Private key decrypt", Algorithm parameter to "RSA", "No padding", and "SHA-1", and the Algorithm parameter should be left as is (using the defaults setting). Regrading the Key parameter, students should select "Set Key", then click on the "Load Key" to load the "Private Key" and set the Mode parameter to ECB. Regarding the Input/Output parameter, students should choose the encrypted message file that he/she received from the partner for input and choose a new one for the output file. Finally, students should hit GO. The original decrypted message will be found in the working folder as a plain text file. Partners should collect and zip their working folder and send it to their instructor for grading.

3 Assignment Summary

Summary	This assignment explains the use of the Asymmetric Encryption Cryptography. Students will communicate secretly using Public Key algorithm using a software called Kryptos.
Topics	Applying Asymmetric Encryption algorithm using Kryptos.
Audience	This assignment is for a course on Cybersecurity, and for students with no prior knowledge with Cryptography.
Difficulty	This is a low to intermediate level assignment, taking 1 week for a (CS1 - CS2) student.
Strengths	This is a very interesting assignment for students who are interesting to learn about Cybersecurity. It is easy and fun to do. Students like this assignment very much.
Weaknesses	The learning curve for using the Kryptos software. Students need some time to learn it and become familiar with it.
Dependencies	Students should understand the general concepts of Asymmetric Cryptography before they can do this assignment.
Variants	Because there are quite a few encryption software, Kryptos was chosen as a nice and easy tool for students to implement what they learned about Public Key Cryptography.

Building A Memory Reading Circuit*

Nifty Assignment

Bin Peng
Department of Computer Science and Information Systems
Park University
Parkville, MO 64152
bpeng@park.edu

1 Assignment Description

This assignment is for an undergraduate Computer Architecture course. Students are required to build a sequential circuit to simulate memory reading. Before this assignment, students have studied combinational circuits and sequential circuits such as adders, multiplexers, memory cells, and ripple binary counters. Students have also practiced building simple combinational and sequential circuits in Logisim[1], a well-known graphic tool for designing and simulating logic circuits. The purpose of the exercise is to combine knowledge of digital circuit components and ROM to create a memory reading circuit, which is an essential part of a computer.

This circuit will read a block of data out of a memory device and display the data on a user terminal. The block of data will be a string of ASCII characters. The characters are hardcoded into the memory device during design time. Memory addresses will be generated automatically using a counter. The students design proper control logic to have each character read out of the memory device sequentially and displayed. The memory device is simulated using a Logisim ROM component. The user terminal is simulated with a TeleTYpewriter (TTY) in Logisim.

Students are given the following instructions:

1. Pick a string with at least 35 characters (could be more than one sentence). The character count includes spaces and punctuation.
2. Decide the minimal number of address bits (n) needed to address each of those characters. n will be the data bits attribute of the counter as well as the address bit width attribute for the ROM component.

3. Encode the string in ASCII and express the values in hexadecimal.
4. Enter those ASCII values in the ROM component in Logisim. If there are still unfilled cells in the ROM, fill in space characters.
5. Build the circuit in Logisim. The basic connection is from the n-bit counter to the ROM, and then to the TTY. The data output of the ROM device needs to go through a bit selector to only let the lower 7 bits into the TTY. Both the counter and the TTY need a clock signal. The string should be displayed in the TTY character by character.
6. Add an On/Off button to turn on/off the circuit. This is to simulate the power button of a computer. The button is off initially. The ROM and the TTY should be enabled when the button is poked once (On) and then disabled when the button is poked again (Off). When a TTY is disabled, it should not accept any data for display (though previously displayed characters will remain on screen). Additional wiring plus basic gates like AND, NOT gates may be used.
7. For extra credits, set up the circuit so it will clear the TTY display and reset the counter back to 0 with some action(s) of the On/Off button.

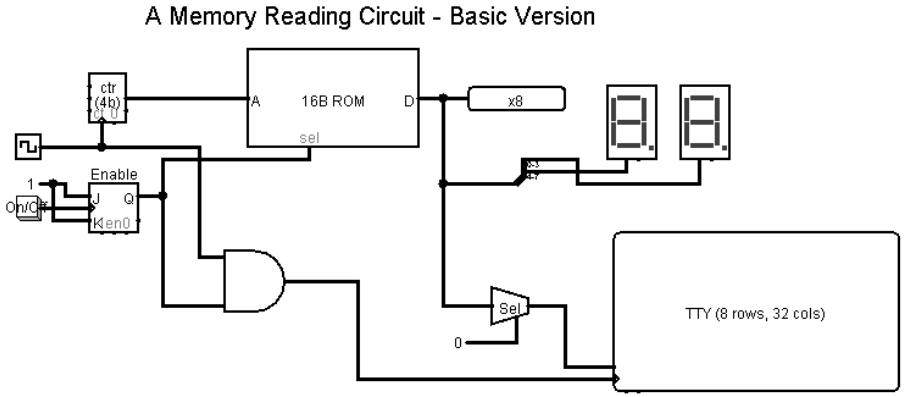


Figure 1: A Memory Reading Circuit

Figure 1 shows the basic circuit. This circuit uses those Logisim components: counter (the component with a label of "ctr" in Figure 1), ROM, bit selector (the component with a label of "Sel"), and TTY. A few optional components are included in this circuit: an 8-bit output data pin (the component labelled "x8" which connects to the ROM) to display the ROM output in binary, two Hex Digit displays to display the ROM output in Hexadecimal, and a splitter (the component directly connects to the two Hex digital display) to

split the 8-bit output of the ROM into the higher 4-bit and the lower 4-bit for the Hex displays. Students are expected to use the Logisim Documentation to figure out how to use those components. Students have used those components in previous assignments: clock, constant, button, JK flip-flop, and basic gates.

2 Additional Ideas

For an advanced class, the instructions may be further simplified to let students figure out the interconnection.

The Logisim counter component may be replaced by a synchronous counter built by students. In the author's class students built a 4-bit synchronous counter in a previous assignment. That circuit could be extended to an n-bit synchronous counter and then used in this circuit as a sub-circuit.

This assignment may be altered to build a memory writing circuit. With the Logisim keyboard component, the circuit may accept user input, echo display on a TTY and store the input in a RAM device.

References

[1] Logisim. `http://www.cburch.com/logisim/index.html`. Retrieved November 25, 2018.

Geospatial Data Handling*

Nifty Assignment

Saty Raghavachary
School of Engineering
University of Southern California
Los Angeles, CA 90089
saty@usc.edu

1 Description

The goal of the assignment is to get students to think about geospatial data - how they could generate such data using their smartphones, represent the data using a simple, capable, and widely-used, text-based 'KML' file format, and easily visualize the data on Google Earth. Further, it shows students how they can process (query) the data they collected (eg. find the nearest location in their sampled data, to another one; compute the enclosing 'convex hull' for their data collection), and visualize the query results and the hull, also using KML and Google Earth. This exercise empowers students to be able to handle any other type of geospatial data in the future, and even use their own plotting schemes (colors, symbols ...) to visualize spatial data; it also provides a platform for them to perform more powerful spatial data queries in the future, via library calls in Python or R.

The assignment appears to satisfy the various items in the 'niftyness' checklist that is on the Nifty Assignments site (`http://nifty.stanford.edu/`) - specifically, the assignment is:

- 'fun', visual
- doable by a variety of students
- extensible by students in multiple ways
- easily adoptable by instructors
- modifiable by instructors

I have been using this assignment for 4 years (8 terms) now, it is consistently a student favorite. Students like that it is hands-on, useful/relevant, engaging and open-ended, so they enjoy working on it.

2 Materials

There were 4 types of files given for their project. Some types consisting of multiple files. For instance ProviderName.csv has a csv file for each provider, so there could be dozens of this type of file.

- smartphone (to obtain latitude/longitude ('lat,long') coordinates, at sampled locations)
- a simple KML starter file (XML format) that is used to represent the sampled (lat,long) coordinates
- Google Earth
- free spatial database software (Oracle+Oracle Spatial, Postgres+PostGIS, MySQL, or QGIS)

3 Metadata

Summary	The assignment asks students to sample spatial data using their smartphone (locations of campus buildings, eateries etc.), visualize the collected data on Google Earth, write simple spatial queries, and visualize the query results also on Google Earth. The spatial queries consist of the following: computing the convex hull of the sampled locations; finding the three nearest neighbors of a chosen location.
Topics	Spatial data collection, representation, querying, SQL, visualization.
Audience	CS1/CS2 students. CS1 students can do data collection and visualization, and do simplified query processing without needing database software, and results visualization; CS2 students would use a spatial database program (eg. Postgres) for the query part, using simple SQL syntax.
Difficulty	Easy (CS1)/intermediate(CS2). The difficulty level would vary (easy vs medium difficulty), depending on whether spatial querying (eg. finding the nearest neighbor of a given location) is done 'by hand', or by using database software.
Strengths	The assignment would provide students, exposure to the vast, colorful world of geospatial information - it does so using simple, active-learning steps that turn the students into data creators, analyzers and also consumers. It is an easy on-ramp to the exciting, relevant and useful field of data science.
Weaknesses	The assignment is specifically focused on geospatial data, some students (and instructors) might not have a fondness for this type of data [or about data processing].
Dependencies	None, beyond what the assignment asks to download and install (Google Earth, possibly Postgres - both are free, multi-platform and easy to install).
Variants	After doing the assignment, interested students can use this as a starting point to do much more - connect to a cloud database, create interactivity for end users, write a smartphone app that does similar queries, collect much more data and do more complex queries, utilize open-source map software for more advanced visualizations, etc. Likewise, instructors can modify the assignment in multiple ways, eg. have students analyze locations of supermarkets and fast-food restaurants, study the distribution of schools in surrounding communities, do other types of visualizations, make the data collection process simpler via a button click, etc.

An IoT Assignment Sequence*

Nifty Assignments

Bill Siever
Computer and Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
bsiever@gmail.com

Internet-of-Things (IoT) is still an emerging area with few formal curricular materials. Developing IoT assignments can be challenging for a variety of reasons: a) it spans many topics; b) students may not be prepared for the variety of technologies used; and c) students may not be prepared to develop complex systems. This presentation will detail an assignment sequence used in a lower division IoT application development course. The course is accessible to sophomores who have taken both CS1 and an introduction to computer engineering. It exposes students to IoT application development via modules on six major facets of IoT applications: 1) Intro. to IoT and user interface (UI) principals, 2) UI Behavior, 3) embedded systems, 4) cloud services, 5) multi-device interactions, and 6) mobile app development. Each module has multiple in-class exercises and one out-of-class assignment. The assignment structure presented here may be useful for other IoT courses/projects, software engineering courses, or capstone projects.

The approach is based on refinements from multiple offerings of the course and by existing literature, especially [1, 2]. Classes meet twice a week. The first weekly session is an active learning exercises, where students work collaboratively on new content. The second is devoted to short lectures, quizzes, or assignment assessment.

The assignment sequence was intended to provide a model for end-to-end development of a real-life IoT application. The assignments focus on two smart home applications: a smart light is developed during in-class exercises and a IoT garage door is developed via out-of-class assignments. The course culminates in a project intended to review the end-to-end development process.

Module	Topics	In-Class Work	Assignment
IoT Intro. & UI Principals	Design Thinking User-Stories, Paper Prototypes, HTML+CSS, Responsive Design	Smart Light UI	Garage Door UI
UI Behavior	Objects, Event-Driven Prog., Listener Pattern, JavaScript	Simulated Smart Light	Simulated Garage Door
Embedded	Embedded Dev., State Machines, Timers/Async, IoT Safety, C++	Non-IoT RGB Light	Non-IoT Garage Door Control
Cloud	Cloud Connectivity, Publish-Subscribe, HTTP Protocol, Cloud Services	IoT RGB Light, Cloud Services	IoT Garage Door
Device-to-Device	Sequence Diagrams, More Cloud Connectivity	Cloud Services	IoT Garage Door & Smart Remote
Mobile-Apps	Responsive Design, Cordova Framework	RGB Light App	Garage Door App

References

[1] Valerie Galluzzi, Carlotta A. Berry, and Yosi Shibberu. A Multidisciplinary Pilot Course on the Internet of Things: Curriculum Development Using Lean Startup Principles. In *ASEE Annual Conference & Exposition*, Columbus, OH, 2017.

[2] Linda M Laird and Nicholas S Bowen. A new software engineering undergraduate program supporting the internet of things (iot) and cyber-physical systems (cps). In *2016 ASEE Annual Conference & Exposition*, number 10.18260/p.26192, New Orleans, Louisiana, June 2016. ASEE Conferences.

Blended Courses in Computer Science and Information Systems Education: Adapting to Changing Educational Methods and Needs*

Panel Discussion

Charles Badami, Denise Case, Nathan Eloe, Aziz Fellah, Doug Hawley, Charles Hoot, Diana Linville
School of Computer Science and Information Sciences
Northwest Missouri State University
Maryville, MO 64468
{*cbadami, case, nathane, afellah, hawley, hoot, dianar*}*@nwmissouri.edu*

Education is constantly evolving, requiring the application of new instruction delivery methods so institutions remain competitive in the changing academic landscape. Blended courses, a mechanism in which courses have reduced face to face time while requiring more out of classroom work by students, are one move towards a more asynchronous educational environment. This panel will discuss the implementation of blended versions of a variety of courses with differing mixes of content type from professional development, programming, and theory. Discussion will center around the activities that were successful across a number of courses, techniques that were effective in some classes and not others, and whether a blended format can be successful for courses of all types (technical, theoretical, and professional), across students of all levels.

Challenges of Mentoring Graduate Directed Projects: Profession-Based Learning Through Collaboration*

Panel Discussion

Ajay Bandi, Denise Case, Nathan Eloe, Aziz Fellah
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{ajay, dcase, nathane, afellah}@nwmissouri.edu

Developing teamwork and communication skills in a capstone course (Graduate Directed Projects), and collaborating with real-world clients are essential components of a profession-based learning framework. However, finding real-world clients with suitable projects is a challenging task for educators. This panel will discuss the obstacles and benefits of collaboration with real-world clients, share experiences, and reflect upon issues from different perspectives. Special attention will be given to collaboration with real-world clients, running effective client meetings, assessing individual contribution in group work, and providing hands-on experience with the latest tools and technologies by following agile software methodologies. In addition, the panel will address issues in teaching such courses.

Papers of the 12th Annual CCSC Southwestern Conference

March 22nd-23rd, 2019
Stanford University
Stanford, CA

Welcome to the 2019 CCSC Southwestern Conference

Welcome to the 12th CCSC Southwest Region Conference, held at Stanford University March 22nd-23rd, 2019. We are delighted to be hosting the conference at Stanford for a second time. We want to welcome both veteran and first-time attendees to the conference. We have a strong and varied program planned, but I have always felt that the best part of CCSC is the opportunity to get to know colleagues in our own backyards and foster new community connections that can lead to future collaboration, peer mentoring, career development, and friendship.

We are proud to feature an outstanding collection of keynote speakers this year. Dan Garcia of UC Berkeley will give the opening keynote address. Jessie Wusthoff, the Director of Diversity and Inclusion at Clover Health will share her expertise on disability and accessibility, and help faculty advance their understanding and practice of accessibility beyond simple legal compliance. Prof. Leo Porter of UC San Diego will highlight research results on active learning in the computer science classroom, and give concrete, readily adoptable suggestions for faculty who want to increase student engagement and learning. Dr. Mike Erlinger of Harvey Mudd College will share wisdom and lessons learned from his years at the National Science Foundation, and give practical advice for faculty on how to utilize NSF programs and funding to enhance their research and teaching. We are also pleased to welcome Wesley Chun of Google, who will lead a hands-on tutorial on the use of cloud computing in education. The rest of our program includes two paper sessions, lightning talks, "birds of a feather" group discussions, a student poster session, and a tour of the beautiful Rodin sculpture garden on Stanford campus.

We received 18 research paper submissions this year, and accepted 3 of them, for an acceptance rate of 17%. Thank you to the dedicated reviewers who allow us to retain a rigorous and selective process that ensures high-quality publications.

This conference could not be done without those reviewers, and the other volunteers on the conference committee. Megan Thomas managed the review process as Papers Chair. Diba Mizra, Youwen Ouyang, Leo Port, Paul Cao, and Rick Covington each chaired other aspects of the conference. All of us were organized by our indefatigable Region Chair Mike Doherty.

Cynthia Lee
Stanford University
Conference Chair

2019 CCSC Southwestern Conference Committee

Cynthia Lee, Conference/Panels/Tutorials Chair ..... Stanford University
Megan Thomas, Papers Chair ..... California State University, Stanislaus
Diba Mirza, Authors Chair ..... University of California, Santa Barbara
Youwen Ouyang, Posters Chair ..... California State University, San Marcos
Leo Porter, Speakers Chair ..... University of California, San Diego
Paul Cao, Lightning Talk Chair ..... University of California, San Diego
Rick Covington, Partner's Chair ..... California State University, Northridge

Regional Board — 2019 CCSC Southwestern Region

Michael Doherty, Region Chair ..... University of the Pacific
Dean Nevins, Treasurer/Registrar ..... Santa Barbara City College
Bryan Dixon, Regional Representative .... California State University, Chico
Angelo Kyrilov, Webmaster ..... University of California, Merced
Colleen Lewis, Past Region Chair ..... Harvey Mudd College

Reviewers — 2019 CCSC Southwestern Conference

Stephanie August ..... Loyola Marymount University, Los Angeles, CA
Bridget Benson ..... California Poly State University, San Luis Obispo, CA
Kevin Buell ..... Garmin
Bruce DeBruhl ..... California Poly State University, San Luis Obispo, CA
Bryan Dixon ..... California State University Chico, Chico, CA
Zachary Dodds ..... Harvey Mudd College, Claremont, CA
Michael Doherty ..... University of the Pacific, Stockton, CA
Jeffrey Hemmes ..... Air Force Space Command
Angelo Kyrilov ..... University of California, Merced, Merced, CA
Colleen Lewis ..... Harvey Mudd College, Claremont, CA
Ed Lindoo ..... Regis University, Denver, CO
Nakai McAddis ..... Northern Arizona University, Flagstaff, AZ
Diba Mirza ..... University of California, Santa Barbara, Santa Barbara, CA
Niema Moshiri ..... University of California, San Diego, San Diego, CA
Dean Nevins ..... Santa Barbara City College, Santa Barbara, CA
Muath Obaidat ..... City University of New York, New York City, NY
Youwen Ouyang ... California State University San Marcos, San Marcos, CA
Leo Porter ..... University of California, San Diego, San Diego, CA
Megan Thomas ..... California State University Stanislaus, Turlock, CA
Richert Wang ... University of California, Santa Barbara, Santa Barbara, CA
Howard Whitston ..... University of South Alabama, Mobile, AL

Experience Report: Explorable Web Apps to Teach AI to Non-Majors*

Justin Li
Computer Science and Cognitive Science
Occidental College
Los Angeles, CA 90041
justinnhli@oxy.edu

Abstract

We report on the experience of using web apps to teach AI to students with no programming experience. These apps allow students to explore methodological limitations and modeling assumptions, and provide them with algorithmic thinking experience. We conclude with qualitative student feedback and general observations about this approach.

1 Introduction

Computational methods are increasingly important across many disciplines in the physical and social sciences. While many fields simply ask students to visualize the results of machine learning, other fields have a more substantial intersection with CS. Cognitive science students, for example, may be interested in topics such as natural language processing (NLP), human-computer interaction (HCI), and artificial intelligence (AI). Similarly, economics students may look at algorithmic approaches to game theory, and urban planners may want to create agent-based models of sociological phenomenon.

Guest-lecturing to and co-teaching these non-majors present a unique pedagogical challenge. Psychology or economics students may have no experience in programming, and thus lack the prerequisite knowledge to complete traditional CS assignments such as large coding projects. At the same time, these

interdisciplinary collaborations are an opportunity to introduce *computational thinking* [4] to non-majors. This raises the question of what pedagogical approaches could be applied to meet the needs of these non-CS students.

This paper reports on our experience teaching cognitive science students using three custom-built, interactive, "explorable" web apps, including a course catalog prerequisite extractor, an pattern-matching chatbot, and a Bayesian network calculator. We deployed these web apps in co-taught and cross-listed courses between computer science and cognitive science, at both the introductory and advanced levels. The source code for these apps and their associated assignments are available on GitHub[1], with hosted versions on Heroku[2].

These apps present graphical user interface (GUI) that allows the students to engage in algorithmic thinking and problem solving without writing code. These apps not only demonstrate the topic under discussion, but with guided questions, additionally provide three benefits:

1. allow students to discover the limits of a particular approach
2. allow students to examine modeling assumptions
3. provide an algorithmic problem solving experience

Below, we describe how each app operates, a sketch of the associated assignments, and how they provide some of the benefits listed above.

2 Course Catalog Prerequisite Extraction

The prerequisite extraction app was originally created for an upper-level, cross-listed AI course, where only half the students have any CS experience. As part of the module on NLP and information retrieval, students were given a course catalog and asked to extract the prerequisites of each course onto separate lines, as a department code and a course number. Students were allowed to use any programming language of their choice or, for non-technical students, to use the prerequisite extraction app. The app must therefore allow students to manipulate text in a repeatable way without writing code.

To achieve this goal, students use a GUI to specify how the course catalog should be processed (Figure 1). Students can add transformation rules that select and split lines of text, as well as insert, delete, and replace words as necessary. Once the transformations are specified, the app applies them to a copy of the course catalog, and shows the original course descriptions and the

[1] https://github.com/justinnhli/ccsc-sw-2019-apps
[2] https://ccsc-sw-2019-apps.herokuapp.com/

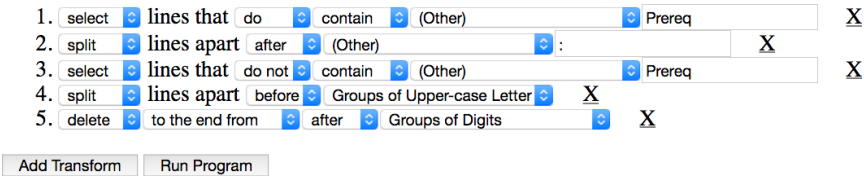


Figure 1: The Course Catalog Prerequisite Extraction app interface.

extracted text side-by-side. This provides immediate feedback for students, who may then modify their transformation rules to fine-tune the extraction.

While students are often quickly able to extract the sentence with prerequisites, the details eventually make perfect extraction difficult. Differences between "CS 101 and 102," "CS 101 and CS 102," and "Computer Science 101, 102" force students to create robust sequences of transformations. Other descriptions such as "any CS 100-level course except CS 130" require additional semantics and data processing to match the required output format. Once students are satisfied with their outputs, they are asked to score their "program" and identify failure cases. The assignment therefore forces students to confront the variability of language, and to experience the challenges both of creating algorithms and of evaluating the performance of an AI.

The interactive app and the questions together demonstrate Benefits 1, 2, and 3. First, despite the seeming generality of the transformations, it is impossible to perfectly extract all prerequisites. Even with a tedious number of transformations to deal with all wording variations, the rules would still fail on negations and other edge cases. The accompanying questions on evaluation further emphasize the limits of such an approach (Benefit 1).

Students are also led to discover misconceptions in their understanding of the domain (Benefit 2). Through this exercise, they realize that course catalogs are not as uniform as they might expect, and even minor variations such as extra commas may cause their programs to fail. For cognitive science students in particular, this may be their first encounter with the difficulty of NLP.

Finally, although students did not write any code, they nonetheless solved a problem with algorithmic/computational thinking (Benefit 3). Much like writing code, students were given the goal of extracting prerequisites, and had to assemble smaller computational building blocks in the appropriate order to achieve the desired result. In essence, the app's GUI provides a domain-specific language for information extraction, without the potential for syntax errors and typos. Through this app, non-CS students can get a sense of the problem solving that programmers engage in.

3 Pattern-Matching Chatbot

The pattern-matching chatbot app was originally created for a guest lecture on the relationship between AI and gender. Mimicking one of the first chatbots, ELIZA, it simply tries to match each received message against a list of patterns, and selects a random response from the first matching pattern. The app was later adapted for a co-taught introductory cognitive science course, as well as advanced courses in both AI and HCI, both of which were cross-listed. Again, the majority of students using the app had no CS experience.

To illustrate the operation of the chatbot, the app provides both a chat window and a textbox where students can edit the patterns and responses. The patterns are indicated by lines that begin with "`@`", while possible responses are indicated by "`%`". The chatbot is also able to match variable text, which are represented by repeated capital letters (e.g. "`XXXXX`") usable in both the patterns and the responses. Whenever the patterns change, they are dynamically loaded into the chatbot, so students have immediate feedback on how the new patterns affect the flow of the conversation.

The accompanying instructions for the app depend on whether the students are absolute beginners or more advanced students. Beginners are simply asked to modify the patterns to remove the mystery of how chatbots might work. For advanced students, this app served as a leaping-off point for deeper discussion of conversational interfaces. By attempting to design a useful chatbot with this limited architecture, students see how semantics, conversational context, and even timing may be necessary for human-level language understanding.

The chatbot app demonstrates Benefits 1 and 3. Although it is obvious that pattern-matching is fragile, students may not be able to articulate the exact scenarios in which the approach breaks down. By placing students in the chat, the app allows students to evaluate the effectiveness of the chatbot (Benefit 1). Advanced students designing chatbots are also forced to consider the effect of patterns being matched in order, and to structure the patterns to provide both specific and generic responses. While the pattern-and-response inputs to the chatbot app are less expressive than that of the prerequisite extraction app, it nonetheless asks students to consider how computational units work together to create the desired output (Benefit 3).

4 Bayesian Network Calculator

The Bayesian network app was created for the cross-listed AI course. Bayesian networks are a graphical model for reasoning about causality and the probability of occurrence of events. In addition to being commonly used in AI and robotics, it has also gained traction as a general modeling tool in the sciences.

This app allows students to visualize a Bayesian network and calculate probabilities. Students enter the causal relationships between events and the conditional probability tables in a textbox, which the app visualizes graphically. Students can then further specify observations of events or to calculate the posterior probability based on those observations.

This app demonstrates Benefits 1 and 2. Since students are asked to create and evaluate a Bayesian model of their choice, they must justify the causal relationships and the conditional probabilities, which often requires data that students do not have (Benefit 1). Students are then prompted for cases where the model gives unintuitive results and to explain the discrepancy — whether the network fails to model the phenomenon due incorrect causality/probabilities, or if it is their intuition that is faulty. These questions and the app together provoke students to critique their models and potentially discover insight about the phenomenon, much as real modelers might (Benefit 2).

5 Qualitative Student Feedback

Student evaluations and feedback over the past three years provide informal evaluation of these interactive apps as pedagogical tools. For example, in the reflection component of the assignment, a student wrote that the prerequisite extraction app provided the *"challenge of figuring out what worked and what didn't work,"* which hints at the algorithmic problem solving required. Another student described how the *"huge amount of variability in how people convey information which makes it difficult to correctly isolate desired information,"* and that as a result, *"coding for information retrieval must be tailored specifically to the corpus."* These comments suggest that the app was effective in provoking examination of the current limits of the NLP. Similarly, a student reflected on their Bayesian network and how they *"didn't think about how those probabilities would interact with each other"*, a failure in their model of the phenomenon that led to discrepancies between the prediction and their intuition. In general, the feedback suggest that these open-ended web apps engaged students in critically assessing the AI techniques. Further study may be able to quantify the effect on student learning compared to other activities.

6 Comments and Conclusion

Interactive pedagogical material is not new. Multiple digital CS textbooks now contain auto-graded coding exercises, which have been shown to improve test scores as compared to static textbooks [1]. Others have applied the lab science paradigm to teach AI, with the goal of increased engagement and providing

experiential learning to students [2].

What differentiates the apps in this paper is the focus on non-CS students. We draw inspiration from "explorable explanations", online documents that combine a narrative with interactive "illustrations" [3]. Such documents emphasize accessibility to a broad audience by preferring graphical interfaces, and often explore a combinatorial space where the user can make self-guided discoveries. Although the web apps presented in this paper are designed with supporting instruction and guided questions for the classroom context, the focus on accessibility is maintained. Explorable explanations have influenced other parts of the design of these web apps. All of the apps have textual inputs (or have inputs directly encoded into the URL), which makes the result easily sharable for collaboration. The open-ended input of text transformation rules and Bayesian network structure aims to make discoveries possible, and it naturally leads to discussions of limitations and modeling assumptions. All three apps are also relatively domain independent, and can by used by non-major students to engage in computational thinking in their domain of expertise.

To the best of our knowledge, neither explorable explanations nor educational web apps have received much discussion in the literature. This paper reported our experience with three different interactive web apps, their design, their use in coursework, and feedback from students hinting at the learning outcomes. In our experience, the presentation of CS content to non-CS students have benefited from use of interactive web apps that allow students to explore CS concepts without writing code. We anticipate that as conventions develop and as supporting frameworks mature, these approaches for non-CS pedagogy will receive more attention and its benefits more formally studied.

References

[1] Alex Daniel Edgcomb *et al.* Student performance improvement using interactive textbooks: A three-university cross-semester analysis. In *2015 ASEE Annual Conference & Exposition*, Seattle, Washington, 2015.

[2] Stephanie Elizabeth August. Enhancing expertise, sociability, and literacy through teaching artificial intelligence as a lab science. In *2012 ASEE Annual Conference & Exposition*, San Antonio, Texas, 2012. ASEE Conferences.

[3] Bret Victor. Explorable explanations. `http://worrydream.com/ExplorableExplanations/`, 2011.

[4] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(1):33–35, 2006.

Investigating University Student Desires and Use of Smartphone Privacy Settings*

Marina Moore[1] and Bruce DeBruhl[2]
[1]Political Science Department
[2]Computer Science and Software Engineering
California Polytechnic State University, San Luis Obispo
{mmoore32,bdebruhl}@calpoly.edu

Abstract

Smartphones provide an abundance of settings to help users control how their data is used. In this paper, we explore what the expressed privacy desires are for college-age smartphone users and whether these are achieved by their settings. To accomplish this, we use a survey and in-person interviews to learn how people feel about their data being shared and how they make decisions about their settings. We find that, in general, people are very concerned with microphone and photo data and less concerned with location data. However, user actions indicate a higher concern with location data than microphones or photo data.

1 Introduction

Smartphones have become a ubiquitous part of people's lives. In our study, almost all respondents (99.3%) report using their smartphone at least once an hour. In addition to being tools for communication, productivity, and entertainment, smartphones have the potential to be tools for surveillance [7]. Developers can use the robust data from smartphones to determine users' whereabouts, habits, and associates. This information allows developers to

personalize the user experience and improve apps [9] but it also has the potential for privacy invasions [9] by corporations and law enforcement.

Fortunately, smartphone users can control how their data is used through their settings [6]. These settings determine which applications have access to certain data, including location, photos, microphone, and contacts. However, these privacy settings are only useful if people understand and use them. Although it is possible to create a phone that is private by design, these concerns are often outweighed by business and usability considerations when creating default settings [4].

Previously, it was found that when it comes to phone security, people are most concerned about losing their device, followed by losing data on their device, followed by abuse [3]. Since privacy focuses primarily on the last of these categories, people tend to make privacy decisions considering the purpose of an app rather than looking at how the data can be abused [6]. Previous research has shown that people's behaviors do not always reflect the attitudes they express [10]. Towards these ends, previous research has attempted to reduce the burden on users to analyze their settings by defining defaults that more closely match people's privacy behavior [6].

In this work, we examine if university students understand their phone permissions and if their permissions align with their preferences. We hypothesize that confusing smartphone settings cause a mismatch between user preferences and behavior. To test this hypothesis we gather data through both a survey and interviews of students. The survey determines phone usage and data sharing preferences, and the interview investigates how people make decisions about their privacy settings.

From our survey and interviews, we make the following observations.

1. When considering photos, contacts, locations, and microphone, people report being least comfortable with observers accessing their microphone or photos.
2. When considering friends, employers, companies, and law enforcement, people report being least comfortable with employers receiving their data.
3. When making decisions about settings, people are the most careful about giving access to their location.

Comparing points 1 and 3 highlights the mismatch we anticipated seeing.

2 Method

We aim to determine if a gap exists between undergraduate student preference and behavior in smartphone usage. We do this with a two-stage experiment.

First, we survey a large group of participants to learn their data sharing preferences. We then interview a subset of these participants to learn about how they make decisions about privacy settings. Throughout this work, we focus on four categories of privacy settings: location, camera, contacts, and microphone. These categories represent a combination of information recorded by the device and information stored on the device. Additionally, location and microphone use are among the most common extraneous permissions requested by apps [8]. To test participants' preferences we also consider four observer groups: friends, employers, corporations, and law enforcement. We choose these observer groups based on common uses and recent privacy cases.

We conduct both the survey and interviews with students at California Polytechnic State University. We choose students as our audience for multiple reasons. Primarily we want a group of people intuitively familiar with smartphones. College students today grew up with smartphones and technology is a ubiquitous part of their school and social lives. As of 2015, the university consisted of 57.2% white students and 85.8% students from California [2]. Although the university demographics are not exceptionally diverse, we believe our results are still illustrative because demographics have not been found to be correlated with privacy preferences [10].

Online Survey - After an informed consent screen, survey participants answered questions about their age, smartphone usage, and data sharing preferences. Age is used to verify if they are in the university age group. For smartphone usage, we ask if the student owns a smartphone, how frequently they use their phone, how often their phone is near them, and what apps they most frequently use.

We state data sharing preference questions in the form "How would you feel if *observer* had access to your *data category*?" The observer categories are close friends, your employer, a company such as Apple or Google, and law enforcement. The data categories are location, camera, contacts, and microphone. It is important to note, that mentioning specific companies may bias the data as people are more likely to trust a named brand [10]. However, we choose Apple and Google as two of the most admired companies [1] who also have massive amounts of consumer data. We collect responses using a Likert scale from very uncomfortable to very comfortable [5].

We advertised the survey to students through popular Facebook pages and email lists. Respondents entered into a drawing for a gift card as a reward for participating.

Interviews - At the end of the survey, we asked participants if they would be willing to participate in a short in-person interview. 42.4% of respondents agreed to participate in the survey, but after filtering for those with iPhones and scheduling we interviewed 13 people (9.4%). We choose to filter

	location	photos	contacts	microphone
friends	comfortable	uncomfortable	comfortable	uncomfortable
employer	uncomfortable	uncomfortable	uncomfortable	uncomfortable
company	uncomfortable	uncomfortable	uncomfortable	uncomfortable
law enforcement	uncomfortable	uncomfortable	uncomfortable	uncomfortable

Figure 1: In this figure, we show observer-category combinations as either comfortable or uncomfortable. Highlighted cells had 75% of responses match the label.

by OS because a majority of respondents, 81.9%, used an iPhone and restricting to a single operating system makes the interviews more homogenous. We note that restricting the population by OS may present bias, however, doing a Mann-Whitney U test on answers for each user shows no meaningful difference between Android and Apple users.

In the interviews, we ask participants to walk through their location, camera, contact, and microphone permissions. We ask them to list which apps have access to their data and how they made decisions to allow these apps to have access to their data. At the end of the interview, we ask participants if they have any additional thoughts about their app permissions and if they plan to change any of their settings.

3 Results

We had 138 respondents to the survey and interviewed 13 of them. Using these responses, we derive insights about how people feel about their data being shared and whether this is appropriately reflected in their settings.

Survey - The 138 respondents are all 18-25 year old students who own smartphones and attend the same university. We test whether smartphone location can be a proxy for participant location by asking how often participants have their phone within 5 feet of their person. 97.9% of respondents report keeping their phone with them regularly, with 66.7% reporting most of the day and 31.2% reporting always.

We consider the participant responses using the nominal categories of comfortable, uncomfortable, and undecided. Using these labels we analyze the observer-category combinations on the survey. In Figure 1, we show comfortable vs uncomfortable with any results agreed on by 75% of respondents highlighted in orange. Using this labeling, it is clear that respondents are uncomfortable with any observers having access to their photos or microphone. Another combination that is consistently uncomfortable is employers having

access to location. The only combinations rated comfortable are friends accessing location and contacts.

By grouping results by data category, we are able to see that people are most uncomfortable with photo and microphone access across observers. Location has the same trend except that people are comfortable with friends having access to their location. We show these results in Figure 2. From this, we determine that people are uncomfortable sharing their photos and giving access to their location and microphone with the exception of friends having access to location. People are more neutral about sharing their contacts, with people being generally comfortable sharing contacts with their friends and generally uncomfortable for other observers.

Interviews - We conducted an extended interview with 13 students. Noticable trends in how people make decisions about settings focus on location. Many students make specific exclusions in their settings based on company reputation.

Most interviewees mention specific features within apps as being the reason they allow apps to have access to a category of data or that they agreed to the pop-up when they felt it was justified. We find that most people allow:

- location for maps or transit;
- photos for social media or messaging;
- contacts for sending money or messaging;
- microphone for social media, video recording, or music recognition/tuner apps.

Most respondents think about their location settings the most. Despite this, all respondents have their location services turned on at the operating system level. All of the interviewees partially use Apple's "while using" option. This option allows them to get the features available from location without constantly sharing their location.

A few respondents specifically forbid certain companies from accessing their data. Three respondents do not allow Facebook access to contacts, and 2 do not allow Facebook to see their location. Additionally, people do not allow Google or social media apps to see their location due to mistrust of the brands.

We find that there is some confusion about what some settings mean, especially photo read and write. In the settings, photo access is broken into read and write, but most apps that request photo access request them both at the same time. One person is not sure what these settings mean, but another wishes for more granularity in allowing apps to write photos without having read access.

After going through their settings in detail, some interviewees learned new information. Four people mention some uncertainty as to why certain apps ask

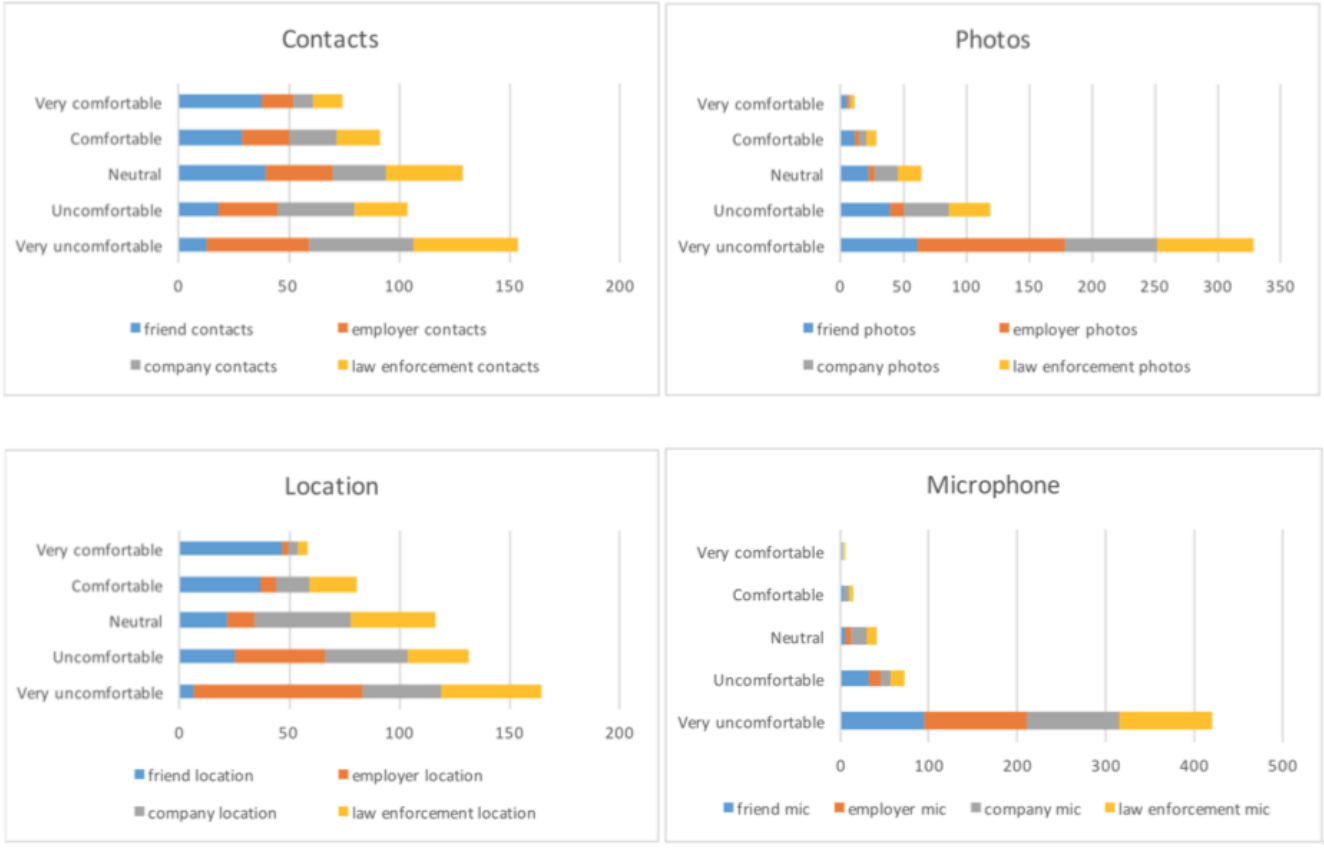


Figure 2: In this figure, we show preferences by data category.

for microphone usage. Our survey indicates that people are most concerned about access to their microphone which is not reflected in how they make decisions about giving microphone access. Three people mentioned specific things they will change after the interview including location settings and deleting an unused app. One person planned to go through their settings again after the interview.

4 Discussion and Conclusion

We find that, in general, students are comfortable sharing their location and contacts with their friends, but are not generally comfortable sharing other data. Despite this, the students interviewed report sharing all categories of data with a variety of companies. This is especially noticeable where many students are uncomfortable sharing their photos and microphone but share them nonetheless.

Future Work - In the future, we propose exploring regional and international differences in privacy behavior by repeating this survey and interview process. Additionally, repeating the study amongst different age groups could demonstrate generational differences in smartphone privacy preferences.

References

[1] World's most admired companies for 2018. `http://fortune.com/worlds-most-admired-companies/`. Accessed: 2018-05-24.

[2] Fact book, 2015.

[3] Marian Harbach, Emanuel von Zezschwitz, Andreas Fichtner, Alexander De Luca, and Matthew Smith. It's a hard lock life: A field study of smartphone (un)locking behavior and risk perception. *Symposium on Usable Privacy and Security*, 2014.

[4] Giovanni Iachello and Jason Hong. End-user privacy in human-computer interaction. *Foundations and Trends in Human-Computer Interaction*, 1, 2007.

[5] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932.

[6] Jialiu Lin, Bin Liu, Norman Sadeh, and Jason I. Hong. Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings. *Symposium on Usable Privacy and Security*, 2014.

[7] Alina Selyukh. The FBI has successfully unlocked the iPhone without apple's help. *NPR*, 2016.

[8] Vincent Taylor, Alastair Beresford, and Ivan Martinovic. There are many apps for that: Quantifying the availability of privacy-preserving apps. *WiSec*, 2017.

[9] Omer Tene and Jules Polonetsky. Big data for all: Privacy and user control in the age of analytics. *NW Journal of Technology and Intellectual Property*, 2012.

[10] Allison Woodruff, Vasyl Pihur, Sunny Consolvo, Lauren Schmidt, Laura Brandimarte, and Alessandro Acquisti. Would a privacy fundamentalist sell their dna for $1000... if nothing bad happened as a result? the westin categories, behavioral intentions, and consequences. *Symposium on Usable Privacy and Security*, 2014.

Less Is More: Assessment and Student Learning in Computer Science Education*

Adamou Fode Made[1], *Abeer Hasan*[2]
Sharon Tuttle[1], *David Tuttle*[1]
[1]*Department of Computer Science*
{adamou.fode, sharon.tuttle, david.tuttle}@humboldt.edu
[2]*Department of Mathematics*
Humboldt State University
{abeer.hasan}@humboldt.edu

Abstract

In this paper, we report on an assessment experiment adding in-class weekly quizzes in an introductory CS 1 course at a public California State University campus. Statistical methods were used to test whether regular assessment of student learning through quizzes leads to better overall grades. Our results suggest that these additional assessments produced negative results in every section in which the quizzes were used.

Key words: Assessments, CS 1, learning improvements, graduation rate, computer science education.

1 Introduction

The last few decades have seen an enormous growth of interest in CS 1 courses as well as the development of many languages at the introductory CS 1 level across higher education. Like many institutions, our institution struggles to find which computing language will produce the best pedagogical outcomes for

an introduction to computer science course. For the past several years, we have used a mixture of DrRacket and C++ for our introductory course, which offers new programmers a smoother introduction to programming concepts before they get bogged down in the heavy syntax of C++. Our approach is based on the "How to Design Programs" textbook [4]. Both types of programming language were taught for precisely 8 weeks.

Although the above mixture has been working relatively well, we have been continually searching for ways to increase student success and improve first-year retention rate. One approach has been to introduce weekly quizzes at the beginning of each weekly lab session, to motivate students to be better prepared. We were hoping that more frequent assessment will keep students on task and thereby improve learning. If students were better prepared for labs, we theorized, they would stay on top of their homework, which would make them better learners and produce better exam outcomes. The weekly quizzes are supposed to provide students with feedback consistently and more accurately, as well as provide a better feel for how the course is going not only for the students but also for the teacher. Weekly quizzes might also provide opportunities to close the gap between current and desired performance.

There is a long-standing scholarship research in the literature on ways to motivate CS students as they begin to learn to program. For example, Peter Brusilovsky and Colin Higgins [5] argued that the use of web learning management system (LMS) allows students to gain understanding and mastery of higher-level programming skills. Others scholars have argued for the necessity of more frequent assessment, see [1], [2] and [3]. In our case, we chose to use an old-fashioned pen-and-paper quiz for several reasons. First, it reduces cheating [5] compared to online take-home assignments. Second, the LMS approach requires having large computing labs which our school does not have available. During these quizzes, we do not want the students to worry about the particulars of a programming environment or be distracted by the computer. Hence we opted for in-class paper quizzes given every week at the beginning of their labs where we can focus on particular syntax and other conceptual matters. Lastly, using an on-paper quiz allows the teacher to give qualitative and personalized feedback rather than rely on automatic generic feedback.

This paper reports on a comparison of student performance across two faculty member's sections of CS 1 with 180 students in the control group and 163 students in the treatment. We aim to answer the following question: "Do weekly quizzes improve students learnings leading them to achieve a higher grade"?

2 Methodology

2.1 Study participants and course description

Our study took place at a campus of the California State University, a public university in the United States of America. The study had two groups, a control group and a treatment group which was given weekly quizzes. Two instructors taught sections of the CS 1 course across multiple years, some sections in the treatment and some in the control group, to reduce the effect of the teacher in the study. Our CS 1 course is open to both a general student audience as well as to first-year students majoring or minoring in CS. The experiment for the control group ran from Fall 2014 through Spring 2016 for four semesters, while the treatment group ran from Fall 2016 through Spring 2018, also for four semesters. The two instructors used very similar teaching techniques, used similar syllabi, and are both established in the CS-education field. Both groups had the same resources available to them, the same written assessments, and equivalent exams. There were 163 students in the control group and 180 students in the treatment group.

Typically, when students finish their lab-activities, they can stay and work on their assignments or they can just submit their work for the day. Each lab session is 110 minutes long, thus there is plenty of time to make room for an extra assignment. Therefore, nothing was removed from the lab-activities to accommodate the weekly quizzes. Each quiz is formatted to take no more than 10 minutes of lab time.

In our CS 1 course, we explore the art and science of problem-solving using the computer as a tool. Course topics include problem-solving, simple expressions and compound expressions, types of data, syntax and semantics, function application, design and implementation, introduction to software testing, Boolean operations, Boolean functions, conditional expressions, and some of the basic structures of computing (sequential, conditional, iterative, and procedural). Students design and implement several programs, first in DrRacket and then transitioning to C++. While studying C++ we also introduce the student to the difference between call by value and call by reference as well as pointers if time permits.

3 Statistical Data Analysis

Figure 1 plot shows no significant difference in the average test scores or the course totals between the two groups. In fact, by exam 2, students in the treatment group (those taking weekly quizzes) start showing either fatigue, or perhaps were overwhelmed with more work than was necessary. By the time

Table 1: Grading weights by assessment category

	Control Group	Treatment Group
Homework Assignments	25%	25%
Lab Exercises	12.5%	10%
Clicker Questions	12.5%	10%
Two In-Class Exams	15%*2 = 30%	15%*2 = 30%
Final Exam	20%	15%

Table 2: Numerical summaries for the control and treatment groups

	Control Group	Treatment Group
Sample Size	163	180
Average Exam Scores	(79.3, 80.8, 75.4)	(78.8, 73.3, 64.9)
Failure Rate (with W's and WU's)	25.8%	24.4 %
Success Rate (with W's and WU's)	74.3%	75.6 %
Drop Out Rate	11.0 %	9.4%
Success Rate (Pass/Fail)	85.9 %	85.6 %

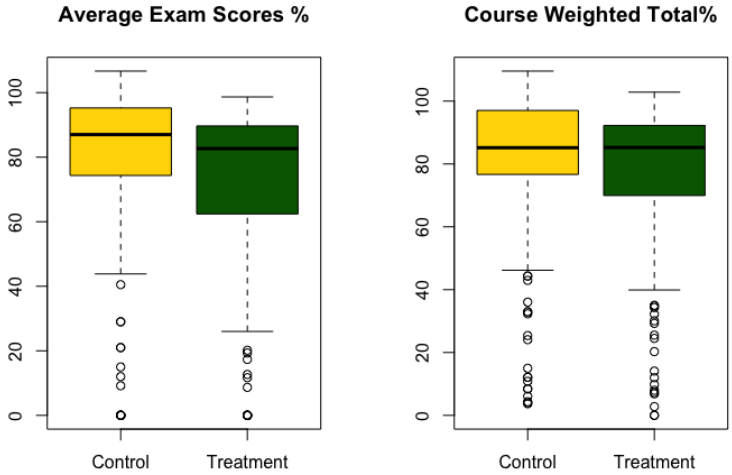


Figure 1: Comparison Average Exam Score and Course Weighted Total

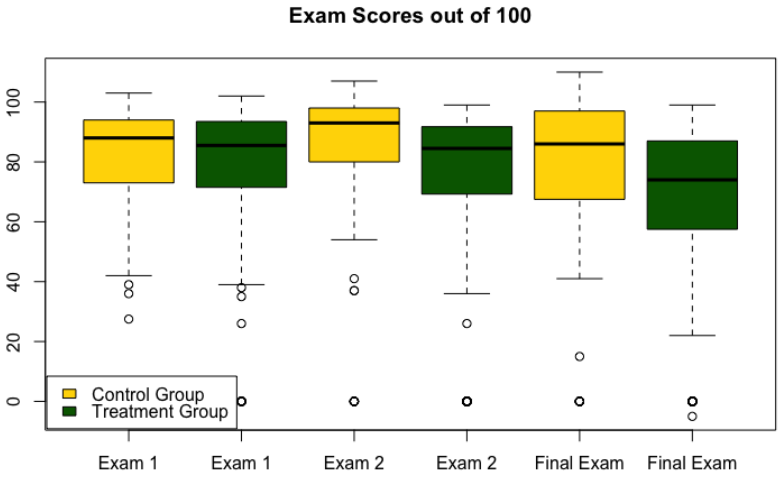


Figure 2: Comparison of Exams for Control and Treatment Group

the students took the final exam, the difference was clear: students who had more assessment were not doing better as shown by the combined test scores, see Figure 2. Moreover, Figure 3 shows that the average lab score for the control group is better than that of the treatment group. Perhaps the extra 10 minutes that was devoted to answering students' questions was more useful than the time the students spent doing the in-class quizzes. There is still an open question to why they performed less on the labs that the quizzes were supposed to help prepare them on.

We compare the medians of the average exam scores for the two groups using Mann-Whitney's Test (Exam scores had an asymmetric distribution and did not pass a normality test. This is why we decided to compare their medians instead of their means). The observed difference in the sample medians was significant with a p-value of 0.0016. The course totals for the two groups were not significantly different (p-value = 0.089). Note that it does not make sense to compare the course total for the two groups because they did not use the same grading rubric. The observed sample difference in the failure rate was insignificant (the 2-sample proportion test gives a p-value of 0.179). There was no significant effect for the section or instructor. The main difference between the two groups was due to the presence and absence of weekly quizzes. All the scores we compared were out of 100, but due to some extra credits points,

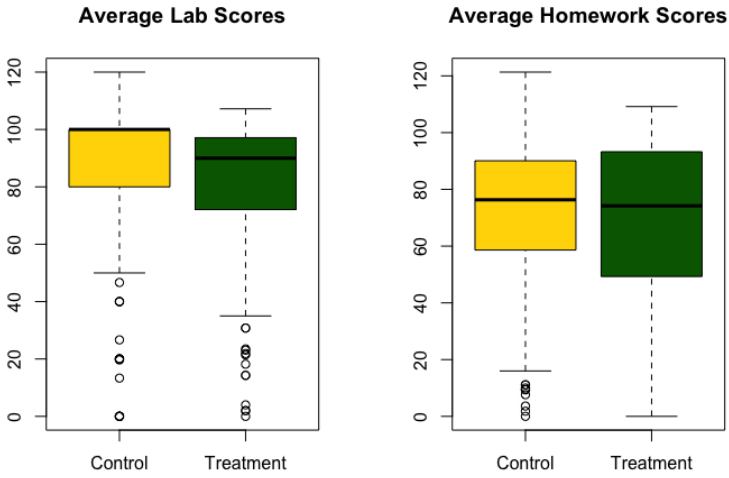


Figure 3: Comparison Average Lab and Homework Scores

some students scored above 100 points .

In the course evaluations, students (fewer than 10) made comment regarding the quizzes. They wanted the quizzes to be graded faster. However, whenever the quizzes were not return within a week, we made sure that the solution to the quiz is available before the next assessment. However, it is worth noting that one student had the following comment: "I prefer to have the assignment that is about the same material as the weekly quiz due before the weekly quiz, because I would have found it easier to remember the code for the quiz if I had used it before hand".

4 Conclusion

Our quantitative analysis provides evidence that having additional assessment beyond homework, lab exercises, TurningPoint technology and three exams in our typical CS 1 course was counter-productive for improving the success rate in the course. However, there may be mitigating factors. It seems highly likely that student fatigue with constant probing may have impacted them negatively. While some students burn out, others improve, and this positive effect also depends on the amount of time spent on homework. It is seems

highly likely that some students did better because they devoted more time to their homework compared to the pressure of taking an assessment. Other students may have put more energy in just doing well on the weekly quiz and neglected their homework. In all of the above cases, instead of increasing the student passing rate, we have created more problems by increasing students' frustrations, which may lead some to abandon the CS major or even education altogether. However, in both cases, other avenues such as tutoring (which is not currently offered for CS1) could help perhaps increase the graduation rate, therefore decreasing the course size with fewer students retaking the course. We believe reducing the drop-out rate is one of the best ways to achieve retention and a higher graduation rate in the CS program in order to meet the ever growing demands in the field of computation. Indeed, this paper provides evidence that Introductory CS is really about inviting and building identity – computational identity – for each individual student. Assessments that, for whatever reason, seem overly onerous or pre-professional, will stunt that growth if it is not already well-established.

References

[1] `https://www.ernweb.com/educational-research-articles/frequent-assessments-student-progress-reduce-pressure-teachers-teach-test/`.

[2] Hasan A. and Ghosh Hajra S. Using oral presentations and cooperative discussions to facilitate learning statistics. *RUME*, 2017. `http://sigmaa.maa.org/rume/crume2017/Abstracts_Files/Papers/55/pdf"`.

[3] Tina Blythe and Associates. *The Teaching for Understanding Guide.* Jossey-Bass, San Fransisco, 1998.

[4] Flatt M. Felleisen M., Findler R. and Krishnamurthy K. *How to Design Programs.* 2018.

[5] Brusilovsky P. and Sosnovsky S. Individualized exercises for self-assessment of programming knowledge: An evaluation of quizpack. *ACM Journal of Educational Resources in Computing*, 5(3), 2005.