The Journal of Computing Sciences in Colleges

Papers of the 21st Annual CCSC Northwestern Conference

October 4th-5th, 2019
Pacific University
Forest Grove, OR

Baochuan Lu, Editor
Southwest Baptist University

Sharon Tuttle, Regional Editor
Humboldt State University

Volume 35, Number 1 **October 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

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
Mercy College

Foreword

The following five CCSC conferences will take place this fall.

Midwestern Conference	October 4-5, 2019 Benedictine University in Lisle, IL
Northwestern Conference	October 4–5, 2019 Pacific University, Forest Grove, OR
Rocky Mountain Conference	October 11-12, 2019 University of Sioux Falls in Sioux Falls, SD
Eastern Conference	October 25-26, 2019 Robert Morris University in Moon Township, PA
Southeastern Conference	October 25-26, 2019 Auburn University in Auburn, AL

The papers and talks cover a wide variety of topics that are current, exciting, and relevant to us as computer science educators. We publish papers and abstracts from the conferences in our JCSC journal. You will get the links to the digital journals in your CCSC membership email. You can also find the journal issues in the ACM digital library and in print on Amazon.

Since this spring we have switched to Latex for final manuscript submission. The transition has been smooth. Authors and regional editors have worked hard to adapt to the change, which made my life a lot easier.

The CCSC board of directors have decided to deposit DOIs for all peer-reviewed papers we publish. With the DOIs others will be able to cite your work in the most accurate and reliable way.

Baochuan Lu
Southwest Baptist University
CCSC Publications Chair

Welcome to the 2019 CCSC Northwestern Conference

The 2019 Northwest Steering Committee is very pleased to welcome everyone to the Twenty First Annual CCSC Northwestern Conference hosted this year by Pacific University in Forest Grove, Oregon.

Many individuals and groups have helped to coordinate and support this year's conference and we want to thank them for all of their time and effort. We especially thank the authors who submitted papers, workshops, and tutorials. This year we have nine papers, four tutorials, student lightning talks, and student posters. The Steering Committee accepted nine out of twelve papers through a double-blind review process. We had colleagues across the region serve as professional reviewers and we recognize their generous efforts in providing time and guidance in the selection of our conference program. We are extremely grateful to have Amy Occhialino, Director of Software Engineering at Intel, begin our conference with her keynote address on Intel's contribution to Open Source Development.

A final thank you goes out to you the attendees whose participation is essential not only to the continuance of conferences such as this, but also for the continued communication and collegiality you provide between all of us involved in the advancement and promotion of our discipline. We hope you enjoy the conference.

Shereen Khoja
Pacific University
Conference Chair

Razvan Alexandru Mezei
Saint Martin's University
Papers Chair

2019 CCSC Northwestern Conference Steering Committee

Regional Board — 2019 CCSC Northwestern Region

Reviewers — 2019 CCSC Northwestern Conference

Carter, Adam	Humboldt State University, Arcata, CA
Chen, Xuguang	Saint Martin's University, Lacey, WA
Davis, Janet	Whitman College, Walla Walla, WA
Guizani, Nadra	Gonzaga University, Spokane, WA
Khoja, Shereen	Pacific University, Forest Grove, OR
Lewis, Robert R.	Washington State University - Tri-Cities, Richland, WA
Mezei, Razvan	Saint Martin's University, Lacey, WA
Smith, Adam	University of Puget Sound, Tacoma, WA
Snider, Brian R.	George Fox University, Newberg, OR

Social Change in Open Source Software[*]

Keynote Speech

Amy R. Occhialino
Director of Software Engineering
Intel

You may know Intel only as a hardware company, and in many ways this is true. Intel's core business is semiconductor design and manufacturing. What may be news to you is that Intel has spent close to two decades working in the open source software community, collaborating on projects that enhance Intel Architecture and advocating for the beauty, elegance, and possibilities that exist within open source software development.

Here are a few facts that you may find interesting:

- Intel is the #1 contributor to Linux kernel, with key industry leaders comprising our software development team.
- Intel is #2 in Linux kernel maintainers.
- Intel has gone from sponsoring 12 open source projects to 200 open source projects, spanning cloud, edge, and device growth, over the past decade.
- Finally, Intel leaders hold over 300 software standards and leadership positions. This equals 10% of Intel's overall software employee workforce.

Open source software development is an enormous commitment and investment at Intel. This feeds into my second point of why Social Change in Open Source Software is an area where I am passionate.

Leading in the future will require a wide range of perspectives, backgrounds, and ideas to effectively solve the world's toughest challenges. We at Intel have the momentum, and the vision, to shape technology's future. And our role is expanding in an increasingly data-driven world, with significant prospects for growth. But we are bound to fail if we do not demonstrate a commitment

[*]Copyright is held by the author/owner.

and passion to increasing and achieving full representative diversity within the open source software industry.

Our success will be threatened in three ways:

- Market Failure: Without diversity-fueled creativity, innovation is stifled. The same perspective and thoughts are generated and reinforced, preventing new solutions from emerging.
- Customer Failure: We will lose customers because we do not listen to them, engage with them, understand them, and learn from them in a full perspective of ways.
- Talent Failure: We will lose top talent because individuals with diverse backgrounds feel out of place in our culture and environment.

Intel is committed to achieving an inclusive and diverse environment in the open source software community and will lead conversations on how the industry can best achieve this vision. Change does not happen all at once, it's incremental, one person doing one thing every day. I personally increase diversity in my own teams through my hiring practices. I have spent a decade creating support systems through my leadership of women groups, and I actively advocate for social change within Intel and my tech community. Open source software gives us the potential conditions, but we must actively engage with it, and monitor it, for it to be what we want it to be.

Biography: Amy Occhialino has spent 20 years in high tech, primarily working for Intel Corporation. She started as a process engineer focused on chemical engineering in the late '90s and spent 18+ years working in all aspects of Intel's semi-conductor manufacturing, from high-volume lean manufacturing to silicon design. In 2017, Amy took the leap to the other side of Intel and began working in open source software. She is currently a Director of Software Engineering, designing and delivering a real-time operating system called Zephyr (https://www.zephyrproject.org/). Amy is the head of the board of directors that oversees the open source governance and strategic direction for the project, hosted at The Linux Foundation. She has also spent over a decade founding and leading women's organizations within Intel, advocating for inclusion and equity. In her personal life, Amy lives in North Portland with her husband and 8-year old twin boys, who currently believe that they are budding Timbers players, and is the PTA president of her awesome local James John Elementary School.

Computer Science in High School: Identifying and Addressing Common Barriers[*]

Yolanda J. Reimer
Department of Computer Science
University of Montana
`yolanda.reimer@umontana.edu`

Abstract

Improving participation and diversity in computer science is an issue that continues to demand the attention of the educational research community, and providing better opportunities to high school students is critical in helping to resolve it. One bottleneck to improving CS offerings at the high school level is the lack of qualified teachers, but there are numerous additional barriers that must also be acknowledged and dealt with. This paper documents the results of a focus group activity whereby we engaged a cohort of high school teachers to better understand what those common barriers are and how we might begin to address them. We learned that teachers face serious limitations in student numbers, student interest, scheduling, support from administrators, and access to resources. We studied the ways in which high school students are drawn into computer science and discuss the recommendations teachers have for expanding on those efforts as well. The broader impact of this work is that it will help teachers and schools across the nation recognize similar issues in their own communities and develop new strategies for alleviating them.

1 Introduction

Improving participation and diversity in computer science (CS) is an issue that continues to demand the attention of the educational research community.

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Many believe that the place to broaden student access to CS is in high school and by providing better experiences at that point in a student's education, more will opt to study CS in college and ultimately choose a computing related career path (e.g., [5]). Research data bears this out by indicating that students who take computer science courses in high school are six times more likely to major in it in college, with women being ten times more likely [3].

As with much of the nation, computer science opportunities in Montana high schools remains low despite a strong market. Based on data compiled by code.org [3], in Montana:

- There are open computing jobs at a rate of 1.5 times the average demand;
- The salary for a computing occupation is about 50% higher than state average;
- Only 40% of all public high schools teach computer science.

A significant bottleneck to offering more CS classes in high schools is the lack of qualified teachers. One approach to dealing with this is providing professional development opportunities and resources, which our NSF CS10K grant has allowed us to do at the statewide level beginning in the summer of 2017. To-date we have trained thirty Montana high school teachers in the Joy and Beauty of Computing (JBC) curriculum [2], and fourteen in Mobile Computer Science Principles (Mobile CSP) [4]. While working with this cohort of teachers over the past two years, many of who returned from one summer to the next, we began to understand that they face numerous other challenges in offering CS at their high schools beyond becoming proficient in new areas of computational thinking and programming.

This paper documents barriers that high school teachers from around the state regularly encounter, including the difficulty of acquiring new skills for a subject area they have not been formally trained in. During a fall 2018 weekend workshop with eleven teacher participants who attended from communities ranging in size from 200 to over 70,000, we engaged in a focus group activity to understand these barriers and discuss possible ways to address them. Much of what we learned from our teacher cohort parallels findings of other related research, particularly in regard to access and understanding of computer science [7, 8]. In our study we also sought to learn more about how students in these high schools become engaged in computer science, and what recommendations teachers have for expanding on those efforts as well. The broader impact of what we report is that teachers and schools across the nation, whether rural or otherwise, will recognize similar issues in their own communities, and will develop some new strategies for alleviating them.

2 Methodology

In November 2018, we held a weekend workshop for high school teachers who participated previously in one of our summer CS professional development classes [6]. Eleven teachers from locations around the state attended. A primary goal of this workshop was to keep the community of educators meeting regularly so that we could continue to share new topics, skills, ideas, and experiences. On the second day of the meeting, we arranged for a focus group activity to understand more about the barriers these teachers face in offering computer science classes in their schools, and ways in which we might help address those barriers.

The focus group activity was organized as follows, including intended time allotments for each step:

1. After an introductory talk about the activity, the group is divided into two smaller groups for phase 2. Each group is moved to separate rooms or corners. (3-5 minutes)
2. Individuals write answers on post-it notes for each of the questions posed (see below) and paste them up on the wall under headings that identify the questions (for each question there is a large sheet of paper on the wall to receive the post-its). (5-10 minutes.)
3. Each of the questions is discussed with the group to clarify what people meant and how many people agree with each of the points. Facilitator adds notes on the wall as needed to clarify any handwriting issues, amplify notes based on the discussion, re-categorizes notes into themes as necessary, and annotates points that received strong group agreement or conversely were the subject of disagreement, etc. (10-15 minutes.)
4. Each facilitator makes photos of the resulting wall to preserve their group's notes.
5. Entire group reconvenes for a final discussion. Facilitators bring their wall sheets to the main room so all sets of answers to each question can be viewed and discussed together. Further clarifications or notes are added to the wall or otherwise recorded as needed to capture useful ideas from the whole group discussion. (15-30 minutes).

Each focus group was asked to answer the following questions as per the procedure outlined above:

1. What have been the major barriers you've faced in offering Computer Science classes in your school?
2. How could this CS10K project help you address those barriers?
3. How have you or your school publicized or "marketed" computer science classes, to invite and encourage students to participate?

4. What are some good ways to help students of different backgrounds to engage with computer science as a potential college major or career choice?
5. How could we make the CS10K workshops or the educational materials and resources more useful? Things we could add or change?

3 Barriers and Ways to Address Them

During the focus group activity, teachers were first asked about barriers to offering computer science classes in their schools. Their comments are grouped thematically and discussed below. The follow-on question of how best to address these barriers resulted in numerous suggestions that are also described within related themes.

3.1 Limited numbers of students in the school, limited student interest, and scheduling issues that reduce the number of students who can participate in a computer science elective course.

The largest number of comments from teachers fall under this theme, marking it as the biggest barrier we identified. Some aspects of this issue are shared by all high schools – such as finding time in already busy schedules to fit in computer science courses that compete with other electives. Rural communities, however, feel the additional strain of small school sizes and competing among a limited number of students who might even consider taking CS.

During the group discussion portion of the focus group activity, teachers noted that if student interest in computer science can be improved, it can drive class availability (scheduling), especially if requests are made in spring while schedules are being created for the following year. An 8-period day could allow for more elective class offerings, or figuring out ways to offer computer science as an independent study might also be a way to alleviate scheduling issues and bring more students into CS.

3.2 Limited support from administrators, other stakeholders.

The second most frequently mentioned barrier is that school administrators and counselors need be made aware of what's available and what computer science is; they need to understand the benefit of coding as a basic skill for all students and one that can lay the foundation for many career choices; and they need to offer more support for these classes. One teacher also suggested that there needs to be community support for changing perceptions about CS.

To address this problem, participants recommended providing enhanced information to school administrators and stakeholders about the importance of computer science education. There has to be an increase in the awareness and understanding of superintendents, school boards, administrators, counselors, students, and communities about the importance of CS education. Some suggested that this might be achieved via targeted letters or meeting presentations, and that national trends and job statistics could be included in informational materials.

3.3 Limited number of qualified teachers.

The third major barrier mentioned is the lack of qualified teachers willing and able to offer CS classes. During group discussion, participants said that other routes to becoming certified to teach computer science in high schools must become available. They also noted that teachers with degrees in other areas, such as business, may not really be qualified to teach computer science.

To help address this issue, teachers suggested looking to the Office of Public Instruction to make changes that would encourage student participation and increase the number of available, qualified computer science teachers. Teacher participants also appreciated and wanted continued professional development opportunities such as the one we offer. One noted that by "training us, then we spread the word, generate enthusiasm."

3.4 Limited access to hardware, software, online resources, curriculum materials.

Even if all other barriers are resolved, limited access to necessary computing resources remains an issue. While this problem is perhaps felt more acutely in rural communities, it is also true in many of the larger towns and cities across the state as well.

To address this issue and assist schools in widening their access to resources, participants heavily promoted the idea of using online tools to share teaching and recruiting strategies, knowledge, and curriculums. They also wanted enhanced collaboration opportunities and noted the possibility of applying for grants to help with hardware and software needs. Many teachers acknowledge that resource sharing has to be inexpensive for it to be practical.

4 Engaging Students in Computer Science Education

To determine how students find out about computer science opportunities in their schools, we asked teachers how those classes are publicized or marketed. Many teachers mentioned that they have individual conversations with students or give presentations in classes, at club meetings, pep rallies, etc. Written and video advertising to students and families is also incorporated in the form of blogs, Facebook, newsletters, hallway displays, and letters. Some teachers use the Hour of Code event [1] to promote programming. Assistance is frequently sought from school counselors and flexible requirements and scheduling is appreciated. Word of mouth by students currently taking CS courses also helps communications.

In terms of recommending ways to help students of different backgrounds better engage with computer science as a potential college major or career choice, themes from focus group comments emerged as follows:

4.1 Provide guest speakers, field trips, job shadowing, experiences with computer science people and applications in the community.

The most frequent recommendation for engaging more students from different backgrounds in CS was inviting guest speakers to come talk with students. Perhaps the most effective guest speakers would have backgrounds in higher education, culturally-relevant areas, and local businesses that are tech related. Others mentioned providing opportunities for students to visit companies, job shadow, and engage in internships. Breaking down possible stereotypes by introducing students to the multiple career opportunities available to computer scientists is also key. One suggestion was to connect college CS students with high school students in a mentoring type relationship.

4.2 Change course requirements or course components.

Teachers felt strongly that computer classes should be made a requirement in high school, perhaps instead of or combined with keyboarding/word processing classes. Others noted that coding could be integrated into other existing classes. Participants noted that the earlier students can be engaged the better, including at the elementary and middle school levels, and that professional development opportunities for K8 teachers should be made available as well.

Lastly, some suggestions for improving student engagement involved highlighting national networked events and organizations that promote coding. One teacher said the "hour of code" could be a school event, perhaps with pizza, and another wanted to find ways to promote women in coding. Teachers also thought that if hardware and software resources could be provided to schools and students with low budgets and access, that would also help spark interest.

5 Discussion and Future Work

After offering multiple professional development (PD) classes and workshops for high school teachers from Montana, we recognize the value of this experience for teachers and believe real progress is being made towards addressing many of the barriers discussed during our recent focus group activity. In particular, the sense of community we are building along with sharing resources and experiences extends well beyond teaching new CS skills. Participants place high value on becoming part of the network of fellow teachers who they can learn from and reach out to long after the PD classes are over.

We also realize, however, that more work needs to be done to help teachers with the additional challenges they face integrating CS in their schools. Both major public universities in the state now offer a CS teaching minor where one did not exist a couple of years ago. We continue to work with high schools on dual credit options for the CS classes they are able to offer. Being able to earn college credit provides significant motivation for students and now, in Montana, the first six dual enrollment credits are free, making this option even more attractive to students. Providing credits for high school teachers that take a professional development class is also very helpful. Many teachers use these earned credits for lane changes (i.e., moving to a higher level of education) and to increase their salary. As a direct result of their feedback, we also printed out and sent completion certificates to all teachers that attended our most recent workshop. We did this so they could help educate and explain their participation in our PD to local administrators and other teachers.

Our research group will continue offering professional development opportunities to teachers around the state for the near future and we have recently secured additional funding to do so. The initial cohort of teachers who started with us in 2017 has expressed interest in continuing their education by tackling additional topics within CS. Our immediate plan is to extend our Python course to include specialized units such as *Python programming with Raspberry Pis*, to integrate more advanced topics like data structures, and to reach out to middle school teachers so we can start the training even earlier.

6 Acknowledgements

Thank you to Michael Coe and David Levine for helping organize and facilitate the focus group activity describe in this paper, and all the high school teachers who participated and offered such sound advice. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1639841. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] Hour of Code. `https://hourofcode.com/us`, retrieved May 13, 2019.

[2] Joy and beauty of computing. `http://ou.montana.edu/t2cs10k/jbc/lectures/index.html`, retrieved May 10, 2019.

[3] Support K-12 computer science education in Montana. `https://code.org/advocacy/state-facts/MT.pdf`, retrieved May 15, 2019.

[4] Mobile CSP, 2018. `https://course.mobilecsp.org/mobilecsp/`, retrieved May 10, 2019.

[5] J. Cuny. Transforming high school computing: a call to action. *ACM Inroads*, 3(2):32–36, 2012.

[6] Y. J. Reimer, M. Coe, L. M. Blank, and J. Braun. Effects of professional development on programming knowledge and self-efficacy. In *Proceedings of the IEEE Frontiers in Education Conference*, FIE, 2018.

[7] C. Stephenson, A. Derbenwick Miller, C. Alvarado, L. Barker, V. Barr, T. Camp, C. Frieze, C. Lewis, E. Cannon Mindell, L. Limbird, D. Richardson, M. Sahami, E. Villa, H. Walker, and S. Zweben. *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions.* ACM, New York, NY, 2018.

[8] J. Wang, H. Hong, J. Ravitz, and S. H. Moghadam. Landscape of K-12 computer science education in the U.S.: Perceptions, access, and barriers. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 645–650, 2016.

LIBRE-ary, an Open-Source, Distributed Digital Archiving System[*]

Ben Glick, Jens Mache
Lewis & Clark College
Portland, OR
{glick,jmache}@lclark.edu

Abstract

As the amount of data we rely on every day has increased exponentially, the problem of digital archiving is becoming more and more important. People increasingly depend on having ready and easy access to the data they need to carry out their everyday lives. With this increased focus on data in everyday life comes a drastic increase in the amount of data people are responsible for. A corollary to this increasing dependence and scale is a problem we would define as digital clutter. Much like physical clutter, digital clutter can lead to confusion, complex data-management strategies, and often, unfortunate loss of important data and documents. In this paper, we present LIBRE-ary, our solution to the digital clutter and archiving problem.

1 Introduction, Related Work, and Motivation

As the world becomes more and more data-dependent, new problems of digital object storage have arisen. A corollary to this increasing dependence and scale is a problem we would define as digital clutter. Much like physical clutter, digital clutter can lead to confusion, complex data-management strategies, and often, unfortunate loss of important data and documents. In this paper, we present LIBRE-ary, our solution to the digital clutter and archiving problem.

One of these new data-related problems is what we call the "digital archive problem." People and organizations have a need to keep track of a huge amount

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

of data and in very specific ways [8]. They are often required by data management plans to keep specific versions of specific objects or certain objects may simply be very important. It is easier now than it ever has been to lose track of the data we depend on every day.

There has been a significant amount of work on the problem of computer backup systems [5, 6, 13, 12], and there has been some work on special, single-purpose digital archives [14, 3, 16, 15], but surprisingly little work has been done on a truly general-purpose digital archive system. This project is able to utilize some of the advancements of other digital backup systems which have been created and operated for some time now, through the digital object storage adapter outlined in the "distributed architecture" section of this paper. We believe that LIBRE-ary is unique among existing digital archive systems in that it is general purpose, completely configurable, fully distributed, sufficiently fault-tolerant, and scalable.

1.1 A Note on the Name

The name LIBRE-ary is not (yet) an acronym. It's inspired by the word Library (as the product is both meant to serve as a library and also was designed initially for use in a library), and incorporate the word "libre," meaning "freedom," which is also a common word used in the open-source community. This word is incorporated because of how important the open-source vision is to the design, implementation, use, and future of LIBRE-ary. The name LIBRE-ary will be used as follows: the noun LIBRE-ary refers to the project itself, while when we refer to "a LIBRE-ary," we are referring to an instance of LIBRE-ary. For example, we may refer to files as being stored in "a LIBRE-ary," while we would discuss design of LIBRE-ary itself.

1.2 Archiving vs Backups

The problems of digital archiving and digital backups are often erroneously conflated. Digital backups provide a solution for the problems that occur when a person's computer is damaged or lost [7]. Digital backups provide "insurance" against hard drive failures but do not help at all with the problem of digital clutter. In fact, digital backups increase the amount of digital clutter that users are responsible for managing. Often, even people who have extensive backups of their devices can still lose important data after their devices fail, simply because navigating all of the backups can be incredibly difficult. Creating digital backups is the process of indiscriminately making copies of every single file [7]. We define archiving, on the other hand, as a completely different problem. Building a digital archive, similar to building a physical archive, requires storing very specific documents along with metadata for those documents, so that archivists always know exactly what they have in their archive, where it's stored, and how secure it is. Building a digital archive can be extremely valuable in the situation where a user is either creating data or otherwise depends on having access to the same specific data. While backups focus on indiscriminately keeping many copies of many different files and even entire systems, digital archives focus on saving only important files, and keeping close track of how and where the files are stored, as well as keeping track of how to quickly recover them when copies are lost.

One example of a situation where digital backups are simply not enough to solve the problems at hand is in libraries. Libraries often have projects that require not just keeping many copies of digital objects but also require keeping complex indices on what the digital objects represent, what other objects they may be related to, where and how they were generated, and how important they are. These challenges cannot be solved by digital backup systems. Instead of simply keeping many copies of anything, it is important when digitally archiving to keep many copies stored in many different locations along with an index of why each object is important and where it came from. With that feature set being considered, we can then move on to thinking about how users might want to interact with an archive system.

2 Design

The key to LIBRE-ary's architecture is twofold: first, there is an approach of generality and abstraction that is able to make LIBRE-ary sufficiently applicable to many situations dependent on digital archives, and second, LIBRE-ary is designed to be inherently distributed and fault-tolerant, in order to ensure reliability and resilience. Additionally, we have designed LIBRE-ary to be both configurable and accessible. It is intended to be able to be used by people with any level of computational experience, intuitive for users of all experience levels - from users with only knowledge of basic, GUI-based computer use, to experienced programmers and other power-users. It is designed to be able to store any kind of digital object, taking into account any necessary considerations which may influence the way said objects are stored.

2.1 Distributed Architecture

LIBRE-ary's architecture is inherently distributed [11]. There is no component of LIBRE-ary which is meant to only run on a single machine. In theory, an instance of LIBRE-ary could be both fully distributed and fully redundant, with a copy of each component each running in concert on separate devices connected over a network. This distributed design is reminiscent of Google's Kubernetes container orchestration system [1]. Each component of LIBRE-ary is designed by contract [4]: there's an interface which instances of each component must conform to, but there're no specific requirements for how each component should operate. The main components of LIBRE-ary are as follows: the metadata manager and associated databases, the various user interface elements (known as "endpoints"), the scheduler, the agent, and a configurable and extensible list of file storage adapters. Each component has a single responsibility and is designed by contract to interact nicely with other components in order to carry out transparent distributed archiving. An architecture diagram of the LIBRE-ary can be found in Figure 1.

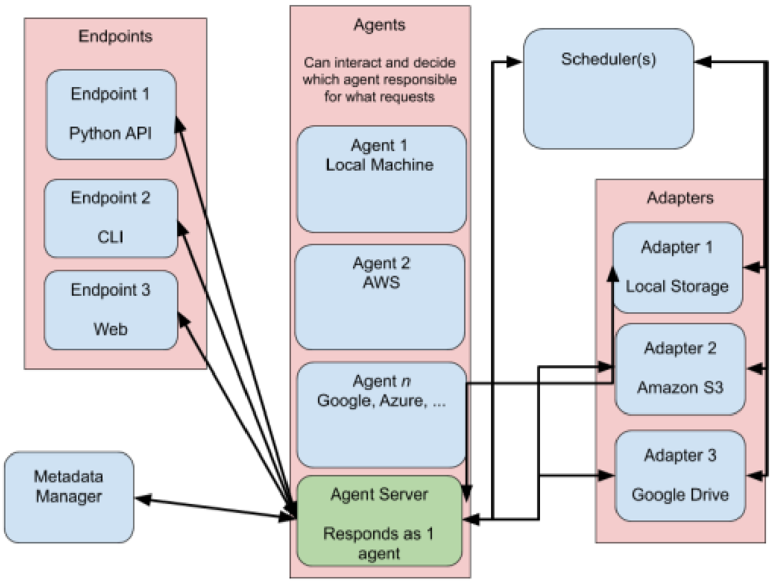


Figure 1: LIBRE-ary architecture including all required components.

The LIBRE-ary agent is the main "manager" of the application. It is responsible for processing user rules, managing configuration, storing metadata, and interacting with object storage adapters. The agent is the only part of LIBRE-ary which must always be running. It is constantly monitoring and synchronizing various shards of the metadata database, managing scheduled data integrity checks, and ensuring that the correct copies of the correct objects are all stored in the right places. It is also responsible for starting the ingestion process of a new digital object into the database. The ingestion process is outlined further in the "Archive and Retrieval Process" section. The agent is designed to be able to operate in concert with other agents, though this has never been tested to our knowledge. When running with multiple agents distributed across several computers, an instance of LIBRE-ary will be able to continue running through the failure of one of the instances. When a digital object is ingested, the agent is responsible for deciding how to best store the digital object, taking into account users' specified options and available resources.

The metadata manager is responsible for keeping track of what is stored in the instance of LIBRE-ary and where those digital objects are stored and can be retrieved. When first starting this project, we knew that we would need to store metadata about each item in the archive, and so we initially thought to use a structured relational database. However, we realized that the metadata database would have to be equally as resilient as the data stored in the archive itself. We decided it was necessary to utilize a fault-tolerant, distributed database to accomplish this. The metadata manager is responsible for orchestrating the distributed nature of the metadata storage. This is user-configurable: users are able to state where they want the metadata to be stored. Any metadata manager backend must be able to support either SQL-style [10] relational querying or key-value querying. Up to this point, only SQL databases have been used, but there is no reason that a different system, like MongoDB or other NoSQL [2] databases couldn't be used. There is only one instance of the metadata manager running in a LIBRE-ary, though it can store the actual metadata in any number of locations. The metadata manager, perhaps counterintuitively, controls not only metadata about the actual digital objects in the LIBRE-ary, but also stores metadata about the LIBRE-ary itself. The main example of this kind of data are the descriptions of digital objects' importance levels. More about these levels is described in the "Archive, Integrity Check, and Retrieval Procedure" section of this paper.

The LIBRE-ary scheduler is, rather intuitively, responsible for scheduling integrity checks, planned backups, and periodically ensuring that required resources are available. It is invoked by the agent, and is told the frequency with which various checks must be run, and is responsible for orchestrating said checks and invoking resources required. If the scheduled checks find something which is not as it should be, the scheduler is responsible for reporting to the agent that it needs to be fixed. The scheduler needs to communicate with the metadata manager, so that it can access information about the levels that digital objects fit into and what tasks need to be scheduled. There can be several schedulers in a LIBRE-ary, each scheduling tasks for either various subsets of the digital objects in the archive, or objects which belong to different criticality levels. If a scheduler ever fails, another scheduler in the LIBRE-ary can take on its role. The number of schedulers in a LIBRE-ary can be chosen by the user.

In order to provide a user interface which is usable by many types of users with varying levels of computational skill and experience, we define the notion of an "endpoint." An endpoint is a generalization of a user interface as a contract [4]. These endpoints can all look and feel very different, but all satisfy the interface which is exposed by the LIBRE-ary agent. LIBRE-ary comes with several endpoints built in, including a python interface, a command-line client, and a simple web interface. These endpoints all behave very differently and are easily accessible to different types of users, but all comply with the LIBRE-ary agent's interface and interact nicely with the agent. Because the agent's interface is an open standard, it is easy to design endpoints which comply with the interface. It is also easy to incorporate similar endpoints into a LIBRE-ary. Endpoints do not need to run on the same machine as any other part of the LIBRE-ary. As long as the endpoint is able to communicate with the agent, whether it's through being on the same system or over the network, the endpoint will be able to carry out its function. Because of this, an endpoint could be used as an ingress controller - the LIBRE-ary runs in a private network, which is connected to a gateway machine on the public network. On that gateway machine is the ingress endpoint. An architecture like this allows for close control of who has access to the data stored in the archive, and could be used to create a secure enclave for protected data.

Finally, and most importantly, each LIBRE-ary has one or more digital object storage adapters. In order to allow for storage of digital objects in as many ways as possible, we provide an open interface for a "digital storage adapter." An adapter is a representation of a data storage provider that LIBRE-ary can interact with. The motivation for providing such an interface is for maximum configurability and extensibility. Instead of directly building in support for a few data storage backends, we wanted to build an interface which can easily be used by developers to build out their own connections to specific data storage providers. Defining this interface also allows us to formalize exactly what our adapters' responsibilities are. We define this contract in terms of "verbs," that the adapter must be able to perform. Specifically, each adapter must have the ability to do four things: store, retrieve, update, or delete any object which is currently stored in said adapter. This contract is very general, on purpose, so as to support a wide variety of different storage services as potential adapters. Adapters can range from storage services like AWS S3 or Google Drive to local storage on a computer to SMB fileshare to SQL databases, and many more options.

The overall architecture of LIBRE-ary is designed to depend on no single computer. Because the entire point of an archive is to ensure long-term stability, it would be a mistake to create a design which is too dependent on any one piece of hardware, when computer hardware is notorious for being prone to failure after just a few years. To solve this problem, we have been inspired by the Kubernetes project, which is a distributed container orchestrator that shares these distributed properties. An unfortunate side effect of this design choice is that it makes it difficult to make complete architecture diagrams of all of the possible arrangements of services. The diagrams in this section represent common use cases.

2.2 Generality of Digital Objects

LIBRE-ary generalizes the concept of a digital object. There may be any number of different types of objects that need to be stored, including but not limited to text documents, images, videos, composite files like TAR or ZIP archives, and binary files. Some of these object types have preferred storage methods. The LIBRE-ary agent is responsible for inferring types of data which is ingested to LIBRE-ary and deciding how to store them. For example, text-based documents can easily have their lengths in lines evaluated. The number of lines in a file can then be saved as an extra integrity check. On the other hand, because binary files do not have a notion of "length," this information is not kept. Another example is with composite files such as archives. As their name suggests, archives store more than one file, by bundling them together. It is possible that one or more object inside the composite file may be more important than another. In this case, the agent is able to treat those files inside the archive differently, storing copies of each file using different digital storage adapters. It is important to note that even with this type inference, LIBRE-ary can still store any digital object, even if it has not encountered that specific digital object type before, because it can treat any unknown digital object as a binary file.

2.3 Archive, Integrity Check, and Retrieval Procedure

The archive and retrieval procedure for LIBRE-ary maximizes resilience and fault tolerance. When a file is first ingested, a "canonical copy" of the file is created. The canonical copy is stored through an adapter which has been specified in configuration to be the "canonical adapter." All objects in the canonical adapter (thus, canonical copies of all objects) are automatically set at the highest configured level of importance, ensuring that they will be checked for integrity as frequently as possible. Checksums [9] are taken of the canonical copy, and saved to the metadata database as part of the ingestion process. Then, copies of the canonical copy are stored through the adapters configured for the importance level of the adapter. All copies have their checksums taken and stored in the metadata database as well. If the checksum of a copy does not match that of the canonical copy, the mismatched copy is deleted and a new copy is created.

After ingestion is over, on a schedule, integrity checks take place. In an integrity check, the following process happens. First, a new checksum of the canonical copy is taken and compared to the checksum on file. If there is a mismatch, a recovery is attempted. The recovery process is to iterate through all copies of a digital object, retrieve all with checksums that match the initially saved canonical checksum, and replace the canonical copy with a copy which has the same checksum as the initial copy. After the canonical copy is checked, the integrity check continues to all other copies of the object in the LIBRE-ary. Essentially the same process is repeated on every copy of the object being checked. During retrievals, the canonical copy is preferred over "satellite" copies, when all other parameters are identical. Upon a failed retrieval, other files stored on the same adapter will be checked for integrity and have retrievals attempted.

In order to make the LIBRE-ary archive process work for a broad variety of applications, we designed a configuration system which allows for various "importance levels" of digital objects. Levels define how important objects are, controlling what adapters they are stored in and how frequently integrity checks are run on them. Levels are completely user-defined, while their enforcement is handled by LIBRE-ary. One common use case we have seen is three importance levels: low, high, and critical. Objects stored at the "Low" level are stored on two adapters (plus a canonical copy), while objects stored at the "high" and "critical" levels are stored in four adapters. Objects at the "low" and "high" levels are checked for integrity weekly, while objects at the "critical" level are checked for integrity daily. Depending on applications, any integrity check frequency or adapter set is possible.

3 Conclusion

As the world becomes more and more data-dependent, new problems of digital object storage have arisen. In this paper, we present LIBRE-ary, our solution to the digital clutter and archiving problem. There has been a significant amount of work on the problem of computer backup systems, and there has been some work on special, single-purpose digital archives, but surprisingly little work has been done on a truly general-purpose digital archive system. We believe that LIBRE-ary is unique among existing digital archive systems in that it is general purpose, completely configurable, fully distributed, sufficiently fault-tolerant, and scalable. The problems of digital archiving and digital backups are often erroneously conflated. Digital backups provide a solution for the problems that occur when a person's computer is damaged or lost. Archiving, on the other hand, is a completely different problem. Building a digital archive, similar to building a physical archive, requires storing very specific documents along with metadata for those documents, so that archivists always know exactly what they have in their archive, where it's stored, and how secure it is.

LIBRE-ary's design has taken an approach of generality and abstraction that is able to make LIBRE-ary sufficiently applicable to many situations dependent on digital archives, and LIBRE-ary is also designed to be inherently distributed and fault-tolerant, in order to ensure reliability and resilience. LIBRE-ary's architecture is inherently distributed. There is no component of LIBRE-ary which is meant to only run on a single machine. In theory, an instance of LIBRE-ary could be both fully distributed and fully redundant, with a copy of each component each running in concert on separate devices connected over a network. This distributed design is reminiscent of Google's Kubernetes container orchestration system. The "distributedness" of the LIBRE-ary design ensures maximal fault tolerance and recoverability. It is designed to depend on no single computer or piece of computer hardware. LIBRE-ary generalizes the concept of a digital object. There may be any number of different types of objects that need to be stored, including but not limited to text documents, images, videos, composite files like TAR or ZIP archives, and binary files. The archive and retrieval procedure for LIBRE-ary maximizes resilience and fault tolerance, and ensures that documents stored in LIBRE-ary are properly replicated and kept up to date. All of these features and design considerations make LIBRE-ary a useful solution to the problem of digital clutter and digital archiving. There is currently an instance of LIBRE-ary running at the library of a liberal arts college, where it helps to ensure library digital assets are kept track of. It is continually being monitored to ensure it runs as it should.

A Appendix A

LIBRE-ary is an open-source project, and the source code is available here: `https://github.com/benhg/libreary`. We encourage people interested in working on LIBRE-ary to fork our repository and submit pull requests.

References

[1] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[2] Jing Han, E. Haihong, Guan Le, and Jian Du. Survey on NoSQL database. In *Sixth international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011.

[3] Shahram Izadi, Abigail J. Sellen, Richard M. Banks, Stuart Taylor, Stephen E. Hodges, and Alex Butler. Archive for physical and digital objects, U.S. Patent 8,199,117, June 12, 2012.

[4] J-M. Jazequel and Bertrand Meyer. Design by contract: The lessons of Ariane. *Computer*, 30(1):129–130, 1997.

[5] Nadav Kedem. Mass storage subsystem and backup arrangement for digital data processing system which permits information to be backed up while host computer (s) continue (s) operating in connection with information stored on mass storage subsystem, U.S. Patent 6,076,148, June 13, 2000.

[6] Gregory Kenley, George Ericson, Richard Fortier, Chuck Holland, Robert Mastors, James Pownell, Tracy Taylor, John Wallace, and Neil Webber. Digital data storage system with improved data migration, U.S. Patent 5,276,867, January 4, 1994.

[7] Catherine C. Marshall, Sara Bly, and Francoise Brun-Cottan. The long term fate of our digital belongings: Toward a service model for personal archives. *Archiving Conference*, 2006(1):25–30, 2006.

[8] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[9] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[10] Michael Stonebraker. SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10–11, 2010.

[11] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms.* Prentice-Hall, 2007.

[12] Keith Walls. File backup system for producing a backup copy of a file which may be updated during backup, U.S. Patent 5,163,148, November 10, 1992.

[13] Kristen J. Webb. Method and system for backing up digital data, U.S. Patent 6,675,177, January 6, 2004.

[14] A. W. Wong, H. K. Huang, Ronald L. Arenson, and Joseph K. Lee. Digital archive system for radiologic images. *Radiographics*, 14(5):1119–1126, 1994.

[15] Albert WK Wong, Ricky K. Taira, and H. K. Huang. Implementation of a digital archive center for a radiology department. *Medical Imaging VI: PACS Design and Evaluation*, 1654:182–191, 1992.

[16] Filip Jay Yeskel. High volume document image archive system and method, U.S. Patent 6,115,509, September 5, 2000.

Design of a Database Graduate Course as a Leveling Class for Non-CS Major Graduate Students[*]

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

Abstract

This paper describes the design of a new graduate computer science course. CSC210 Database Fundamentals is a fundamental course in Saint Martin's University (SMU) for the Bachelor of Science in computer science and Bachelor of Science in information technology. Many applicants to the Master of Science in Computer Science in SMU are from non-computer majors. They are usually required to successfully complete several undergraduate courses as their leveling classes, before officially starting their graduate courses. One of such leveling classes is CSC210. However, such a solution has some limitations during the implementation, so several new graduate courses focusing on those applicants are proposed, one of which is about database fundamentals and primarily in accordance with CSC210. This new graduate course introduces the topics covered in CSC210 plus additional knowledge and skills required in graduate studies. Other than the coursework such as assignments, midterm exams and quizzes, the new design has incorporated a second term project helping students better familiarize themselves with various research methods and skills. The lecture arrangement of this new course is divided into two phases. The first phase applies the traditional teacher-centered method discussing the main topics from CSC210, while the second phase is more similar to a graduate course, focusing on classroom discussion, self-study, and research project. The new course will be

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

tried in the summer of 2019 as a pilot, still using CSC210 as the course number.

1 Introduction

Computing is one of the fastest growing industries in the United States; the demand for qualified computing professionals is high [2, 1]. Recognizing such a high demand for training in computer science, Saint Martin's University created master's program in computer science in April 2018. Like other universities [2, 6, 7, 1, 4], many applications in SMU are from the applicants with non-computer majors. For these applicants, a usual solution is to conditionally accept them, and then require them to successfully complete several designated undergraduate or graduate computer science courses (as leveling classes), before officially starting their master program. According to the graduate academic catalog of SMU [5], one of leveling courses is CSC210 Database Fundamentals.

CSC210 is offered by the Department of Computer Science in Saint Martin's University (SMU), which is a fundamental course for Bachelor of Science in computer science and Bachelor of Science in information technology. The author has taught database-related courses at several universities and taught CSC210 at SMU in 2016 and 2017.

However, according to the students' feedback and the performance of the students who have completed the leveling classes, this solution has some limitations. Thus, several new graduate courses are specifically designed for the graduate students with non-computer majors, one of which is about database fundamentals. These graduate courses are based on the corresponding undergraduate courses for example CSC210, which not only introduce the topics of those undergraduate course but also cover the knowledge and skills that are required in graduate studies.

In this paper, the background and motivation of designing the new graduate course of database fundamentals is in the second section. The third section is about the detail of the course design, such as the topics to be discussed, coursework, and teaching method. A short discussion about this new design is in the fourth section. Finally, a brief summary and future work is provided.

2 Background and Motivation

For the applicants with non-computer majors, a usual solution [2, 6, 7, 1, 4] is to conditionally accept them, and then require them to successfully complete several designated undergraduate or graduate courses as leveling classes.

However, based on our experience, such a solution is not always helpful for the applicants to make up their expertise.

At first, all these applicants to MSCS already have had a bachelor degree in non-computer majors and certain computer-related work experience in their original fields. Therefore, many topics discussed in an undergraduate course (as a leveling class) are often easy or simple for them, and correspondingly cannot better help them to strengthen their education background in computer science.

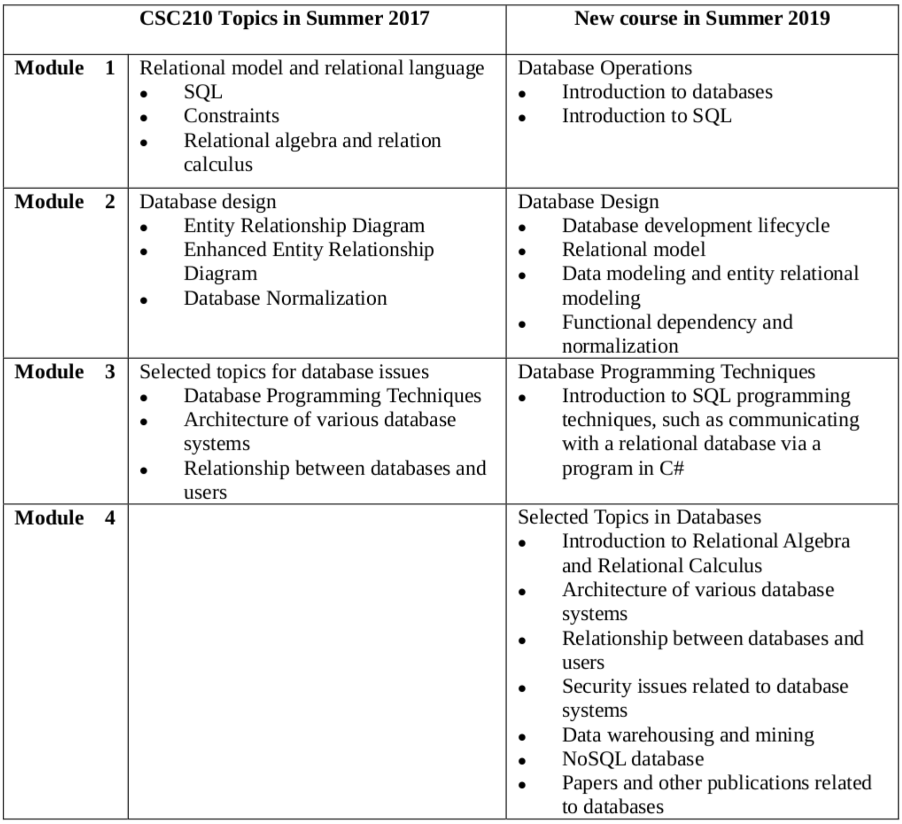
Secondly, the leveling classes must be able to help the students acquire the knowledge, meanwhile developing the skills that the students should possess when they are pursing their master degree, for example, the skills of managing research projects, collecting materials and data, conducting experiments and analyzing the results, writing research report, and introducing research projects in presentations. Because undergraduate courses usually do not fully require or cover these skills, although they can help students learn the lack of computer knowledge, they are not very helpful for the students to acquire the skills needed in MSCS.

In response to the above limitations, we decided to switch to another solution having been adopted by some universities [6, 7], which requires the students to take graduate courses specifically designed as the leveling classes. But, the design of such graduate courses can have some limitations too. For example, the coursework mainly consists of assignments, exams, quizzes, and tests, and lacks the work like term projects [6]. Although such a course design can help the students better understand the techniques and skills discussed in class, they often can not provide the students with an opportunity to comprehensively apply these techniques and skills to a specific application. So, correspondingly, the application experience the students can get from such a course will be limited. Moreover, the grade [6] is primarily determined by the work related to assignments and exams rather that by those required in a graduate course such as group discussion, literature review, research report, and presentation. As the result, because the students have to spend most of time on their assignments and exams so as to keep a high score, the research skills and experience the students can learn will be limited.

Hence, in order to overcome the limitations, we decided to design some graduate courses as level classes having the dual characteristics of undergraduate and graduate courses. They cover the main topics of the corresponding undergraduate course, and meanwhile contain a certain number of topics a graduate course to be said. In their coursework, it focuses more on the work such as term projects, helping students to gain more practical application experience and research experience.

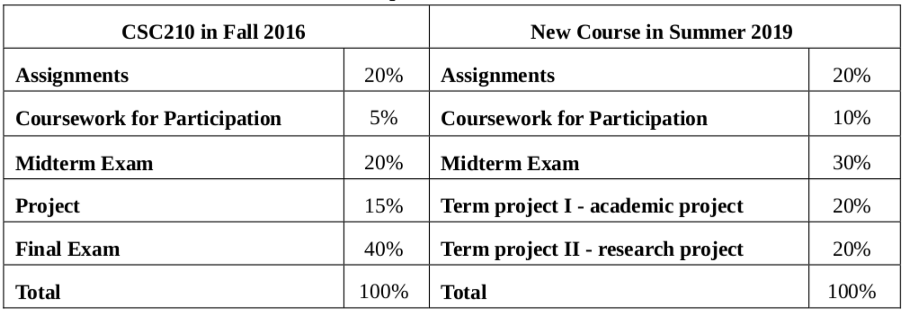
According to the admission requirements [5], if an applicant with a non-computer major has not taken a databases course, then they must complete CSC210 as a leveling class. A sample is listed in Table 1 and Table 2. Because many applicants with non-computer majors already have acquired database operation experience from their work, many topics in this course become too easy for them. Thus, designing a graduate course (CSC5xx) about the fundamentals of database as a leveling class becomes necessary.

Table 1: Topics Covered in CSC210 and in new course



	CSC210 Topics in Summer 2017	**New course in Summer 2019**
Module 1	Relational model and relational language • SQL • Constraints • Relational algebra and relation calculus	Database Operations • Introduction to databases • Introduction to SQL
Module 2	Database design • Entity Relationship Diagram • Enhanced Entity Relationship Diagram • Database Normalization	Database Design • Database development lifecycle • Relational model • Data modeling and entity relational modeling • Functional dependency and normalization
Module 3	Selected topics for database issues • Database Programming Techniques • Architecture of various database systems • Relationship between databases and users	Database Programming Techniques • Introduction to SQL programming techniques, such as communicating with a relational database via a program in C#
Module 4		Selected Topics in Databases • Introduction to Relational Algebra and Relational Calculus • Architecture of various database systems • Relationship between databases and users • Security issues related to database systems • Data warehousing and mining • NoSQL database • Papers and other publications related to databases

Table 2: Sample Coursework and Grade



CSC210 in Fall 2016		New Course in Summer 2019	
Assignments	20%	**Assignments**	20%
Coursework for Participation	5%	**Coursework for Participation**	10%
Midterm Exam	20%	**Midterm Exam**	30%
Project	15%	**Term project I - academic project**	20%
Final Exam	40%	**Term project II - research project**	20%
Total	100%	**Total**	100%

3 Course Design

3.1 General Principle of Course Design

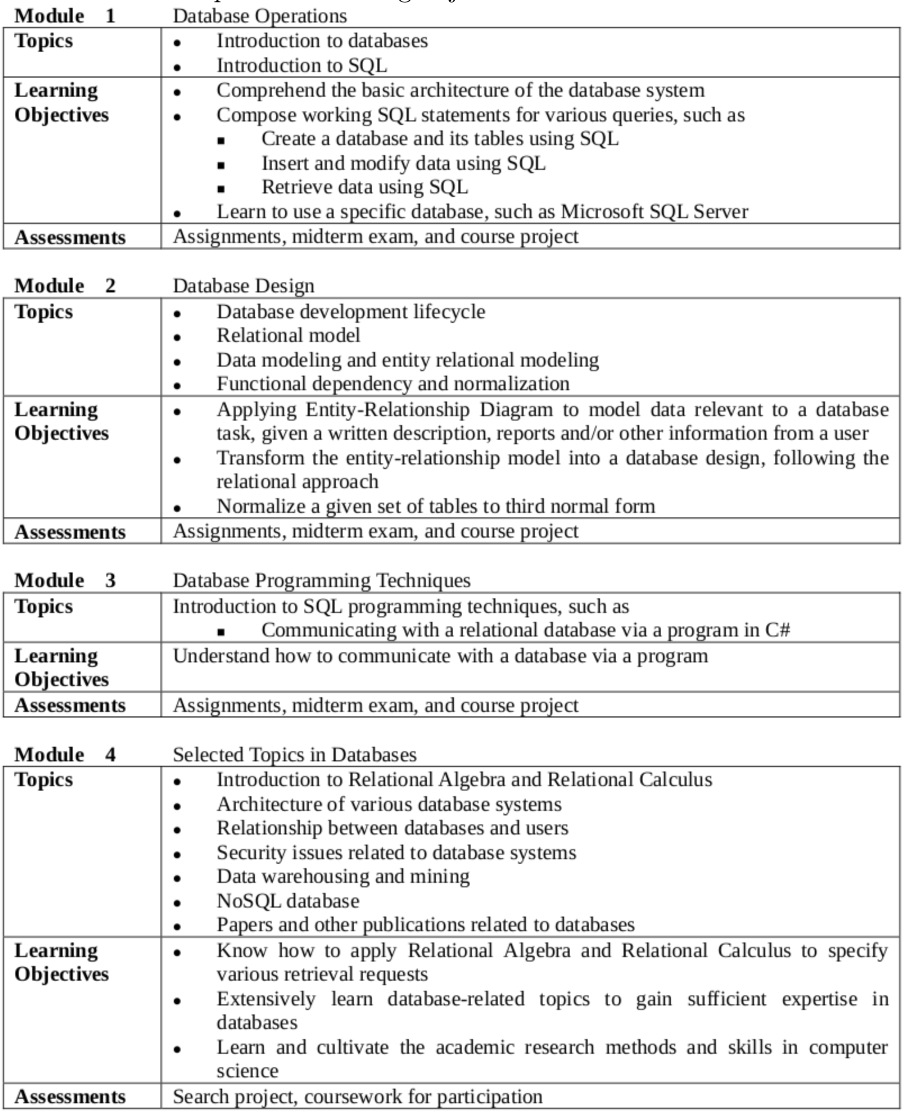
When designing the new database course, the following factors are mainly considered.

- Although it is a graduate course, it must embed the characteristics of both the undergraduate and graduate courses but with a more emphasis on the side of undergraduate.
- In addition to the topics covered in CSC210, topics in a regular graduate course should be embedded.
- Other than the coursework usually used an undergraduate course such as assignments, quizzes, and exams, it should included the work widely used in a graduate course such as independent research projects, group discussions, research reports and presentations.
- During the lecture time, because many students have had database operation experience, the hours assigned to the topics from CSC210 will be compressed so that the remaining hours are allocated to the newly added ones.
- When discussing the topics from CSC210, pedagogic methods is similar to the teacher-centered method, and when working on other topics, the student-centered approach will be applied, such as in the form of group discussions, self-study, research presentations, and so on.

3.2 Topics in Each Module

The topics from CSC210 are primarily scheduled in the first three modules, as shown in Table 3. The new additions are in the last module.

Table 3: Topics and learning objectives covered in summer 2019



Module 1	Database Operations
Topics	• Introduction to databases • Introduction to SQL
Learning Objectives	• Comprehend the basic architecture of the database system • Compose working SQL statements for various queries, such as ▪ Create a database and its tables using SQL ▪ Insert and modify data using SQL ▪ Retrieve data using SQL • Learn to use a specific database, such as Microsoft SQL Server
Assessments	Assignments, midterm exam, and course project

Module 2	Database Design
Topics	• Database development lifecycle • Relational model • Data modeling and entity relational modeling • Functional dependency and normalization
Learning Objectives	• Applying Entity-Relationship Diagram to model data relevant to a database task, given a written description, reports and/or other information from a user • Transform the entity-relationship model into a database design, following the relational approach • Normalize a given set of tables to third normal form
Assessments	Assignments, midterm exam, and course project

Module 3	Database Programming Techniques
Topics	Introduction to SQL programming techniques, such as ▪ Communicating with a relational database via a program in C#
Learning Objectives	Understand how to communicate with a database via a program
Assessments	Assignments, midterm exam, and course project

Module 4	Selected Topics in Databases
Topics	• Introduction to Relational Algebra and Relational Calculus • Architecture of various database systems • Relationship between databases and users • Security issues related to database systems • Data warehousing and mining • NoSQL database • Papers and other publications related to databases
Learning Objectives	• Know how to apply Relational Algebra and Relational Calculus to specify various retrieval requests • Extensively learn database-related topics to gain sufficient expertise in databases • Learn and cultivate the academic research methods and skills in computer science
Assessments	Search project, coursework for participation

Module 1. Database Operations

Because many graduate students with non-computer majors have extensively operated databases before taking the leveling class, the lecture will start from database operations, helping the students further understand the architectures of a relational database system and how to create SQL queries to communicate with a database. Moreover, some concepts such as primary key, foreign key, super key, and relationship among the data in different tables are covered here, which will be applied in the next module to the topics such as entity-relationship diagram (ER diagram) and enhanced entity-relationship diagram (EER diagram), database normalization, and database development.

Module 2. Database Design

This module has three stages. At first, the database development lifecycle and relational model are covered, helping students:

- Understand that a database can not be developed randomly or by directly writing SQL queries.
- Learn the procedure and steps involved when developing a database.

Next, the topics related to relational model, data modeling, entity relational modeling, and especially entity relation diagram are followed, helping the students:

- Understand the meaning and application of each symbol needed to create an ER or EER diagram.
- Learn how to draw an ER or EER diagram for a particular scenario, when designing a database.

Finally, functional dependencies and database normalization are discussed, helping the students:

- Understand the concepts of functional dependencies and database normalization.
- Learn how to normalize a database to the 3rd normal form (3NF).

Module 3. Database Programming Techniques

This module introduces the students how to use a computer program to communicate with a database system, performing the operations such as retrieving information for a database, creating a database, and modify a database. Considering the fact that the students are from non-computer majors, the primary purpose of this module is to help students understand the procedure of database programming rather than learning the skills of database programming.

Module 4. Selected Topics

This module has two stages. Firstly, it provides students an extensive knowledge of databases especially the following aspects:

- Understand the job responsibilities of various personnel working for a database, such as database administrators, database designers, and tool developers.
- Understand the topics related to database architectures, like three-schema architecture, DBMS component modules, database system utilities, and centralized and client/server architectures.
- Know the security issues related to databases.

Then, the students will be provided various papers, books, technical reports and perform literature review about databases, helping students:

- Know the classification of databases and the unique features of each type of database
- Acquiring the experience of operating different database systems
- Carrying out research activities in certain areas of computer science to accumulate research experience and learn various research techniques and skills required for graduate studies.

3.3 Coursework

The coursework is listed in Table 2. As shown in the Table, the final exams have been removed in the new course so that the students can spend more time on their two projects. In addition, quizzes via coursework for participation and written in class are added.

Coursework for Participation

The coursework for participation includes the activities such as mini quizzes, exercises in class, reading assigned papers, attendance check, group discussion and presentation. The size of such a work is small and provided flexibly. The purposes are to evaluate the performance of each student and help students review/understand the basic idea or application of a particular topic. For instance, after explaining the Entity Relationship diagram (ERD), a mini quiz asking the students to write down the name and meaning of each symbol can be provided immediately. After a group discussion in class, a mini presentation can be scheduled.

Assignments and Midterm Exam

The first a few assignments and midterm exam are mainly used together to evaluate the performance of each student working on the topics of database fundamentals. The assignment are close to those in an undergraduate course, focusing on practicing each skill and technique discussed in class, and on the other hand, the midterm exam focuses on the comprehensive application of all techniques and skills practiced in the assignments. The rest of assignments will be more close to graduate assignments, requiring the students, for instance, to read the specified papers or other materials, then conduct research to find the answers, and finally report or summarize their results as the answers.

Term project I - Academic Project

The academic project is semester-long (15 weeks) and completed by 2 to 3 students as a team. It helps the students better understand the techniques and skills discussed in class by comprehensively applying them to a specific application, meanwhile acquiring real working experience in database development. The project usually was either a prototype or if available, a real one from industry. The whole project will be completed in two iterations. Firstly, applying what they have learned in Module 1, the students collect the data, design the database, and created a database in SQL. The techniques in Module 2 such as ER/EER diagrams and database normalization are encouraged but not mandatory. Then, when introducing database design and programming technique, the students revise their databases by:

- Creating or if having created, revising ER/EER diagrams of their database.
- Normalizing their database to 3NF in SQL.
- Creating other documents such as system architecture, user manual, and tutorials.
- Creating a C or PHP program communicating/manipulating the database.

In each iteration, the students need to finish the corresponding paperwork (for example weekly report or progress report). In the end, each group will have a final presentation and report, comprehensively introducing what they done and their achievements in their project.

Term project II - Research Project

Similar to the academic project, the research project is also semester-long and completed by 2 to 3 students as a team. Different from the academic projects, the project is selected from the instructor's research area, but tailored for this class. When the project is employed, the students act as the research assistants and the instructor is their client and advisor. Under the supervision of the instructor, the students manage the whole project and conduct various research activities. The purpose of this project is to help the students learn various research methods and skills useful for their future graduate studies and accumulate practical research experience. Based on the number of students enrolled, several projects will be prepared and provided to the students at the beginning of the semester so that each student team can choose the most interested one.

4 Implementation and Discussion

The new course will be tried in the summer of 2019 as a pilot, using CSC210 as the course number. The effectiveness will be measured by different methods, such as:

Course evaluation

At the end of each semester, students in every class will complete a course evaluation. We will compare the evaluations from this course to the previous evaluations to find out if this design is more helpful to students.

Student survey

At the end of the summer semester, students taking this course will be required to complete a survey specifically drafted for the course design, helping to identify the strengths and weaknesses of the design and how to improve.

Performance in coursework

Some coursework used in previous semester (such as assignments and projects) will be selected and used in this summer. The student's answers will be compared to the previous answers to determine the effectiveness of the course design.

In author's opinion, compared to CSC210 and other graduate courses used as leveling classes [7], the design in this paper can potentially have several advantages.

Firstly, by taking this course, the students can learn the knowledge of computer science (especially database-related knowledge) that they lack. Meanwhile, the course project can provide the students with an opportunity to comprehensively apply a variety of knowledge and skills learned in class to a specific application, thereby helping students gain practical experience in database operations and development. Additionally, when completing the research project, the students can learn the methods and skills needed in graduate program through various research activities, thereby helping them gain some practical research experience and laying a foundation for their research in the future.

Moreover, this approach can contribute to the instructor's career development, closely combining teaching and research. Universities in the primarily undergraduate category are usually smaller in size and offer fewer graduate programs. The faculty in such a university has to teach more courses and often have no research assistants. In this design, when working on the research project, the student can act as research assistants, helping the instructor employ research projects that cannot be done alone and thus providing support to the faculty career development.

5 Summary and Future Work

This paper introduces the design of a new graduate computer science course related to database fundamentals in Saint Martin's University (SMU). Recognizing such the high demand for training in computer science, master's program in computer science (MSCS) was created in SMU in April 2018. Many applicants to MSCS are from non-computer majors, so they are usually required to successfully complete several undergraduate courses as their leveling classes, before officially starting their graduate courses. One of such leveling classes is CSC210 which is a database fundamental course for undergraduate students.

However, such a solution has some limitations during the implementation, so several new graduate courses as leveling classes are proposed, one of which is about database fundamentals and primarily in accordance with CSC210.

This new graduate course not only introduces the topics covered in CSC210 but also has additional knowledge and skills required in graduate studies. Other than the coursework such as assignments, midterm exams and quizzes, a second term project [3] is incorporated helping students better familiarize themselves with various research methods and skills. The lecture arrangement of this new course is divided into two phases. The first phase applies the traditional teacher-centered method discussing the main topics from CSC210, while the second phase is more similar to a graduate course, focusing on classroom discussion, self-study, and research project.

As the future work, the new course will be tried in the summer of 2019 as a pilot, still using CSC210 as the course number during the trail. Then, based on the feedback collected from the students, this new course will be further modified and submitted for approval in the fall of 2019. If approved, it will be officially listed in the SMU's undergraduate academic catalog as a 500-level course and offered in 2020-2021 academic year.

References

[1] University Of South Alabam. Masters program for non-majors. https://www.southalabama.edu/colleges/soc/masters-program-for-non-majors.html.

[2] Kristin Burnham. How to get a Master's in computer science without a CS background, October, 2017. https://www.northeastern.edu/graduate/blog/how-to-get-masters-in-computer-science-without-cs-background/.

[3] Xuguang Chen. Redesign of a senior software engineering course with dual projects. *Journal of Computing Sciences in Colleges*, 33(1):194–201, 2017.

[4] The University of Texas at San Antonio. M.S. degree. https://cs.utsa.edu/students/graduate/masters-degree.

[5] Baylor University. Graduate leveling. https://www.baylor.edu/csd/index.php?id=94628.

[6] Brandeis University. Master of Science in Computer Science for non-majors. https://www.brandeis.edu/computer-science/graduate/12-course-masters.html.

[7] Kennesaw State University. Computer Science, MS. http://catalog.kennesaw.edu/preview_program.php?catoid=39&poid=4925&returnto=3082.

Course Models for Teaching Data Science[*]

Haiyan Cheng[1] *and Tammy VanDeGrift*[2]
[1]*Computer Science Department*
Willamette University
Salem, OR 97301
hcheng@willamette.edu
[2]*Donald P. Shiley School of Engineering*
University of Portland
Portland, OR 97203
vandegri@up.edu

Abstract

This paper describes two models for data science courses offered by computer science programs at two different universities. At both institutions the data science course serves as an elective course for computer science majors. The paper presents course outlines, topics, labs, and projects associated with the courses, so that others can adopt and adapt these models. The course topics and the projects are similar at both institutions. One institution has more frequent lab-based assessments and the other has fewer more open-ended assignments. Both courses have been offered once and the instructors are satisfied with the student learning outcomes.

1 Introduction

Data Scientist was considered the best job in 2018 by Glassdoor's rankings [5]. Preparation for this career comes in several forms. People with mathematics, statistics, and computing backgrounds can get on-the-job experience or

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

complete masters degrees in data science. Another option is to complete data science bootcamps on-line or in person [16]. The job titles vary just as much as the preparation – data analyst, data scientist, data engineer, or sometimes a more general database administrator. While the College of Charleston has offered a degree in Data Science for over ten years [3], other universities are just starting to offer courses and programs in this area. This paper describes recently offered data science courses at two private universities in the Pacific Northwest. Both serve as elective courses for CS majors and minors and are designed to provide exposure to foundational topics in data analysis, statistics, model-building, machine learning, data visualization, and ethics.

2 Data Science Courses

2.1 Background & Development

Both authors developed data science courses to serve as elective courses for computer science majors. Neither computer science program had an existing course in data science or big data analytics. For the purpose of this paper, the institutions will be referred to as UnivA and UnivB. Both UnivA and UnivB are semester-based private institutions on the west coast that offer degrees in computer science. UnivA has around 50 computer science majors across four cohorts. UnivB is slightly larger with 180 computer science majors across four cohorts; The curricula at both institutions are similar with an introduction to computer science course followed by a data structures course. Table 1 contains information about both courses, based on their initial offerings.

Table 1: Data Science Course Information

	UnivA	UnivB
Course Name	Intro to Data Science	Intro to Big Data Analytics
Name in Paper	DS 400	BD 400
Prerequisites	One Junior-level CS course beyond Data Structures	Data Structures & Statistics
No. of Students	32 of max 24	24 of max 25
First Offering	Spring 2018	Spring 2019
Rotation	Every other year	Every other year
Languages	R, Python and Tableau	R

Since BD 400 was offered one year after DS 400, the UnivB faculty member used the UnivA course materials in developing BD 400, in addition to other on-line materials [9, 12, 7]. The author from UnivA has background in computational science and predictive modeling, and already has experience using R, Python and Tableau in research. The author has prepared the course by participating collaborative research projects for a consecutive three summers, working on various data projects including anomaly detection, predictive policing, and DNA hotspots binding prediction. The hands-on collaborative nature of data science projects inspired the author to design and offer a data science course to further explore individual CS student's potential and prepare students better for job opportunities. In order to prepare for the course, the author from UnivB completed on-line courses through EdX for Microsoft's Data Science certificate program [12]. Since that author already knew Python and Python is used in the curriculum at UnivB, the author elected to learn R and use R in BD 400, so that students developed skills in a new programming language. The motivation for offering a course in Big Data Analytics came from UnivB's computer science industrial advisory board; members of the board indicated that more people need exposure to this area.

Table 2 includes topics covered in the courses. The X in the column indicates the topic was covered in the corresponding course. The table, however, does not necessarily indicate the order in which topics were presented. As the reader can see, most topics were consistent across both courses.

The main textbook for DS 400 was *Doing Data Science, Straight Talk from the Frontline* by Cathy O'Neil and Rachel Schutt [13]. Other textbooks for recommended reading in DS 400 include these: [17, 15, 11]. The required textbooks for BD 400 were *Data Science and Big Data Analytics* and *An Introduction to Statistical Learning with Applications in R* [14, 9] with recommended reading as [13]. DS 400 used RStudio for R to perform exploratory data analysis (EDA), Anaconda Jupyter notebook [1] for Python machine learning and Tableau for visualization. BD 400 used RStudio as the development environment for R and the Shiny application platform for data visualization [2].

2.2 Course Assessments

Both instructors focused on student learning through hands-on labs, homework assignments, and projects. Both instructors used similar grading schemes (93+ A, 90-92 A-, 87-90 B+, 83-87 B, 80-83 B-, etc.). Table 3 shows the weights of the course assessments toward the final grades.

2.3 DS 400

The instructor for DS 400 organized the course on a weekly basis for the total of 15 weeks. The lectures are 90 minutes long on Tuesdays and Thursdays. Each lecture has an associated 90 minutes lab. The lecture time include powerpoint presentation, hands-on code demonstration and students interactive code developing, as well as watching videos and general discussions. The instructor

Table 2: Course Topics

	DS 400	BD 400
Intro to Data Science	X	X
Intro to R	X	X
Intro to Python	X	
Intro to Tableau	X	
Statistics & Probability	X	X
Exploratory Data Analysis	X	X
Linear Regression	X	X
Classification	X	X
Logistic Regression	X	X
Decision Trees		X
Unsupervised Learning	X	X
Clustering	X	X
Data Wrangling	X	
Data Visualization	X	X
Platforms: Spark and Hadoop		X
Ethics	X	X

also assigns weekly required reading and exercises, which include background reading for relevant research papers, textbooks, software installation, mini-exercise, tutorial for R, Python and Tableau.

Topics: The class topics were broken down into weekly chunks, with relevant required tasks:

1. Intro to data science, data science life cycle.
2. Exercise on personal data science profile, practice asking questions about data, R intro and exploratory data analysis using R.
3. Probability distributions and data science algorithms.
4. R linear regression and multiple regression through examples.
5. Supervised learning, classification, logistic regression, K Nearest Neighbors, K-mean in R.
6. Hierarchical clustering in R.
7. Tableau introduction and tutorials.
8. Python introduction, environment setup and tutorial.
9. Python data projects case study through examples.
10. Numpy basics
11. Pandas and data wrangling with examples.
12. Pandas plotting and visualization.
13. Pandas generalized models and scikit-learn classification algorithms.

Table 3: Weights and Deliverables

	DS 400	BD 400
Homework/Labs	40% (2 assignments)	21% (6 labs)
Exam or Quizzes	20% (1 midterm)	15% (6 quizzes)
Project	30%	25%
Participation	10%	10%
Research Paper		15%
Personal Data Project		7%
Presentation		7%

14. Data science ethics

Homework: The instructor for DS 400 designed two collaborative open-ended group assignments. Each homework assignment was done in teams of three to five students. Each homework submission includes a written summary report, the source code, and an in-class presentation. Here are the descriptions of the assignments:

1. Exploratory Data Analysis with R, with a focus on various visualizations and pattern finding: Using New York Times ads and clicks information from May 2012, perform exploratory data analysis and make visual and quantitative comparisons. Some guidelines and suggested steps were provided to help students get started. For this homework, eight teams were formed to work on the same dataset.
2. Linear and Multiple Regression with R, learn how to analyze the quality of the models. For this homework, eight teams were formed. This homework has two parts, in part 1, all teams were required to work on the same fake dataset. In part two, three real datasets were provided, team 1, 2, 3 worked on dataset 1, teams 4, 5, 6 worked on dataset 2, and teams 7, 8 worked on dataset 3. For each dataset, a series questions were asked to give students hint on what needs to be done.

Group project brought out students creativity in visualizing data from different angles and encouraged them to explore various interesting facts. The in-class presentation provided students opportunities to learn from others, especially those teams that work on the same datasets. Students were required to form their own team. Each team has a leader who is responsible for allocating, coordinating tasks, submitting deliverables and reporting individual contributions. Different teams were formed for the two homework, so that the students have opportunities to work with various classmates and learn about each other's strength, weakness, as well as personal interests to prepare for their final project team formation.

Project: Students worked in teams on a culminating final project of their interest. Teams chose a relatively large dataset to explore. In doing the project, students learned about the data domain, how the data is organized, conducted exploratory data analysis, generated models, generated a visualization, and communicated the results through a final in-class presentation. Each team is also required to make a poster to present at the university's annual SSRD (Student Scholarship Recognition Day) event.

Each team discussed their proposal and obtain approval from the instructor, who helped each team to set their project goal. Team formation and project proposals were due three weeks prior to the final presentations. A proposal template was provided for students to report the following information:

1. Project title
2. Team members (first person is the leader)
3. Data and its description: source, number of files, format, number of record in each file.
4. Proposed project goal and tentative algorithms and approaches.
5. Research questions planning to answer or hypothesis planning to test.
6. Forecast or model techniques.

Nine final team projects were produced with the contents:

1. Statistical analysis of Movies from 1980-2017
2. Analysis of CS degree conferred in OR and CA
3. Whale identification challenge
4. Chocolate bar ratings
5. Crime analysis of Portland Oregon
6. Federal employee public policy research
7. NCAA college basketball analysis
8. Pokemon data science
9. Analysis of noise complaints in New York City

The UnivB instructor used this project as a model for the project in the BD 400 course. Some differences were that in DS 400 students completed the project with a combination of R, Python and Tableau, then presented the final results through a formal in-class presentation and a poster. In BD 400, only R is used.

Student Presentations: Two 90-minute class sessions were scheduled for students to present homework 1 and 2 results. Critiques, feedback and discussions followed each presentation. The 3-hour final exam time was used for the final project presentation.

2.4 BD 400

The instructor for BD 400 organized the course around six two-week modules, with each module including three lecture days, two lab days, one-half day for student presentations, and one-half day for a culminating quiz. The course met Mondays, Wednesdays, and Fridays for 55 minutes per session. An example two-week block is presented in Table 4. Lectures focused on the theory and algorithms for data science and labs focused on using R packages to apply the algorithms on example datasets. Student presentations focused on contemporary research papers that utilized the techniques studied in the module.

Modules: The class topics were organized into six modules with associated labs, student presentations, and quizzes:

1. Intro to data science, statistics, data life cycle, and programming in R
   (a) Lab: Making projects in RStudio, Exploratory Data Analysis, Plotting, Cleaning Data
2. Data models, linear regression, and multiple regression
   (a) Lab: Regression, Multiple Regression, Non-Linear Transformations
3. Supervised learning, classification, logistic regression, datasets: training, validation, test, K Nearest Neighbors
   (a) Lab: Logistic regression, training models, KNN, gradient descent for coefficients for logistic regression coefficients
4. Tree-based models, Bayes classifier
   (a) Lab: Classification trees, Bayes classifier, Regression trees, Bagging, Boosting, Random Forests
5. Clustering, unsupervised learning, principal component analysis
   (a) Lab: Principal component analysis, k-means, hierarchical clustering
6. Association rules, recommendation systems, data visualization, data presentation, ethics
   (a) Lab: Association itemsets and rules, data visualization with Shiny

Labs and Pre-labs: Students worked with assigned partners to complete the labs. The assigned partners rotated every lab, so each student worked with six different students in the class. Each lab had a pre-lab assignment based on readings that were completed individually and due at the beginning of the first lab session. Most labs were designed from the labs and datasets provided in [9]. The instructor created six to eight checkpoints for the two-session labs and student work was graded on completion of the checkpoints.

Table 4: Two-week Block in BD 400

Monday	Wednesday	Friday
	Quiz on Previous Module / Lecture	Lab Day
Lecture	Student Presentations / Lecture	Lab Day
Lecture / Review	Quiz on Module & Lab Due	

Project: A semester-long project was assigned the first week of the semester. Students chose teammates based on shared interests in domains outside computer science. The eight project topic areas that emerged from students' interests included the National Basketball Association, video game sales, global workforce participation of women, carbon emissions in the USA, free/reduced lunch and the CPI, gun ownership and mortality, grocery store access and income, and mathematical sequences. To ensure that students made progress outside of class, project deadlines happened throughout the semester:

- Team Formation: Upload team member names (due beginning of third week)
- Project Proposal: Upload proposed project domain, dataset the team plans to explore, and a preliminary exploratory data analysis (due beginning of sixth week)
- Project Demos: Present models created from data (due middle of 13th week)
- Project Code and Dashboard: Final code and Shiny dashboard complete (due middle of 15th week)
- Letter: Teams chose a recipient and wrote a letter about the models and data (due middle of 15th week); this was in place of a formal paper to give students a substitute for a formal customer/client who would be interested in the results of the data analysis
- Presentation: Teams did a formal oral presentation during the final exam timeslot

Research Paper: Students completed individual research papers of 2000 to 2500 words about how data and analytics have transformed a domain. The paper included the types of data that are collected, how the data is analyzed, and models built from the data. This was due at the end of the ninth week of the semester. Students chose a wide range of domains: food and agriculture, weather forecasting, sports, military, shopping, and prediction systems.

Personal Data Project: Students completed personal data projects throughout the semester. The assignment asked them to choose a personal, measurable goal and keep a data diary about progress to the goal. For example, they could choose a goal of applying to ten jobs or studying at the library for at least one hour per day or practicing a musical instrument at least three timers per week. A reflection about their goal, the data collected, how collecting and using data helped inform them about personal decisions, and what they learned about themselves was due at the beginning of the 13th week of the semester. Students chose personal goals in exercise habits, practicing music, maintaining a budget, applying for jobs or internships, and calorie consumption.

Student Presentations: Seven 30-minute sessions were scheduled for student presentations that happened every other week. In the middle of each two-week module, a group of three to four students presented results from research papers that used the data modelling techniques in that part of the module. The instructor deliberately chose research papers from authors around the world, so students could expand their understanding of global issues. The instructor chose three to four papers for each student presentation session, so students typically divided the presentation by one paper per presenter. The students were asked to do the presentation, do some active learning activity with the class, and submit a quiz question based on the presentation. After the presentation, the instructor posted students' slides to the course management system, so they had access to this material when studying for quizzes. Example papers from around the world include the following: [10, 8, 4].

3 Results, Conclusions and Proposed Future Changes

Both courses emphasized the placement of data science within the domains of programming skills, math/statistics, and domain knowledge, as shown in Drew Conway's venn diagram [6]. DS 400 included more lectures on statistics, Python and Tableau. Because BD 400 had a prerequisite of a course in statistics and probability, just one lecture reviewed this material. Python and Tableau were not part of the course. Both courses required students to do background research about the domain in which they collected data for projects. The instructors of the courses are satisfied with the first offerings of the courses. UnivB benefited from having materials and resources developed at UnivA.

Considering various students background, the instructor at UnivA designed the course with both R and Python to give students exposure for two languages. Together with statistics background introduction and Tableau software, the instructor feels the course has breadth, but may lack depth for a limited time span. To get students' feedback, the instructor designed an online evaluation form, 10 students responded with the following results. For the time spent between R and Python, five students prefer to spend more time on Python, four students prefer more time on R, and one prefers 50-50 split. For language preference on exploratory data analysis (EDA), five students prefer using Python, three students prefer using R, and two has no strong preference. Eight students prefer spending more class time to learn Tableau, two students prefer not. In terms of specific skill growth in the scales of 0 to 10, (0 indicates a specific skill does not grow, 10 indicates the skill has grown significantly), students reported growth shown in figure 1. Students wish to grow more in the following areas in order of importance: Python, data analysis, computer science, statistics and data visualization. Students also reported the most frequently used learning resources beside class material are: R and Python tutorial, stackoverflow, other books and resources, other data science students and youtube. For struggling contents, students reported statistics background and debugging with R and Python. The answers for whether they have any interest in declaring a data analytic minor if it were offered, five said yes, four said maybe, and one said: "I would, but will likely graduate before I get the opportunity".

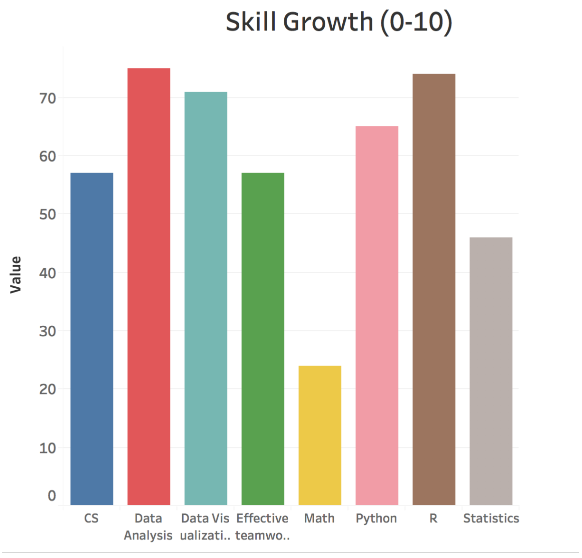


Figure 1: Skill growth (0-10) after taking DS 400 (sum of total 10 students).

After teaching the course for the first time, the instructor at UnivA plans to revise the course in the following ways: split the DS 400 course into a two sequence courses: CS-300 Fundamental of Data Science, and CS-400 Applied Data Science. Both courses will use Python. Statistics and R will be taught in the Mathematics department, Math-200 Statistical learning with R, and will be the pre-requisite for the CS-300. This design will also help to support the planned data science major and minor. With this revision, more smaller homework and exercises in Python will be designed and more advanced content can be added to the upper level course. All students feel the data science course is valuable, especially in helping students to secure summer internships and REU research experience. Five students completed data science-related projects as their senior capstone project. Several students decided to continue Master's degree in data science. This course also resulted two faculty-student collaborative research projects: one with a psychology professor, the other with a public policy professor. The instructor feels data science has its advantages of being hosted in the Liberal Arts setting, where students can combine their diverse background and work across disciplines. The instructor at UnivA also thinks that Ethics in Data Science topics can be developed into a full-fledged course in collaboration with another department.

After the first course offering, the UnivB instructor would change the weights of assessments in the next offering by increasing the weight of the quizzes and reducing the weights of the labs and personal data project. Because the labs and personal data assignment were primarily graded on completion, most students earned high marks in those categories. The quizzes, research paper, and project demonstrated a greater distribution of students skills. Labs were based on those in [9] that included sample code. The instructor provided sample code for labs, but some students thought that was too much scaffolding and wanted less starter code. In future offerings, the labs will include more open-ended coding challenges. Students in the courses completed end-of-semester evaluations about the courses. At UnivB, all twenty-two students that completed the evaluation agreed or strongly agreed that the course was a valuable experience. All twenty-four students showed engagement through consistent attendance and submission of labs, pre-labs, and the project. The instructor for BD 400 invited two recent alumni who work as data scientists to give guest presentations on their work and the techniques and tools they use in their jobs; both presenters encouraged skill development in R and python, data exploration, model-building, and visualizations. One of the students reported that the guest lectures and the class itself provided insight into the data scientist profession and now has data scientist as a career goal.

This paper presented two variants of an elective course in data science. With the rise in data collected by individuals, corporations, and institutions comes the need to manage, use, and protect data. Data Scientist has been named as the best job for three consecutive years [5]. Institutions may want to consider how to prepare students for this career, through elective courses, minors, or designed majors. Because the course blends computing, statistics, and domain knowledge, faculty may want to consider offering a data science course for students across disciplines. Hopefully, these models are helpful for other faculty who want to design data science courses. The reader is welcome to contact the authors for more information and course materials.

References

[1] Anaconda, 2019. https://www.anaconda.com/.

[2] R studio, 2019. https://www.rstudio.com/.

[3] Paul Anderson, James Bowring, Renée McCauley, George Pothering, and Christopher Starr. An undergraduate degree in data science: curriculum and a decade of implementation experience. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 145 – 150, 2015.

[4] Arthur T.E. Capozzi, Giancarlo Ruffo, Viviana Patti, and Cristina Bosco. A data viz platform as a support to study, analyze and understand the hate speech phenomenon. In *Proceedings of the International Conference on Web Studies*, 2018.

[5] Louis Columbus. Data scientist is the best job in America according Glassdoor's 2018 rankings. *Forbes*, January 28 2018. https://www.forbes.com/sites/louiscolumbus/2018/01/29/data-scientist-is-the-best-job-in-america-according-glassdoors-2018-rankings/3dba9e7a5535.

[6] Drew Conway. The data science venn diagram, 2015. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram.

[7] Vasant Dhar. Data science and prediction. *Communications of the ACM*, 56(12):64 – 73, December 2013.

[8] Lasantha Fernando, Sriganesh Lokanathan, Aparna Surendra, and Thavisha Gomez. Predicting population-level socio-economic characteristics using call detail records (CDRs) in Sri Lanka. In *Proceedings of DSMM'18: Data Science for Macro-Modeling with Financial and Economic Datasets*, 2018.

[9] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, 2017. http://www-bcf.usc.edu/ gareth/ISL/index.html.

[10] Preecha Khakham, Narumol Chumuang, and Mahasak Ketcham. Isan dhamma handwritten characters recognition system by using functional trees classifier. In *Proceedings of the 11th International Conference on Signal-Image Technology and Internet-Based Systems*, pages 606 – 612, 2015.

[11] Wes McKinney. *Python for Data Analysis.* O'Reilly, 2012.

[12] Microsoft. Microsoft professional program in data science, 2019. https://www.edx.org/microsoft-professional-program-data-science.

[13] Cathy O'Neil and Rachel Schutt. *Doing Data Science, Straight Talk From the Frontline.* O'Reilly, 2014.

[14] EMC Education Services. *Data Science and Big Data Analytics.* John Wiley and Sons, Inc., 2015.

[15] Jake VanderPla. *Python Data Science Handbook.* O'Reilly, 2016.

[16] Alex Williams. Data science bootcamps, May 30 2019. https://www.coursereport.com/blog/data-science-bootcamps-the-complete-guide.

[17] Nina Zumel and John Mount. *Practical Data Science with R.* Manning, 2014.

The Effect of Peer Tutoring in Reducing Achievement Gaps: A Success Story*

Adamou Fode Made[1], Abeer Hasan[2], Scott Burgess[1], David Tuttle[1], Nick Soetaert[1]
[1]Computer Science Department
{adamou.fode,scott.burgess,david.tuttle,nes239}@humboldt.edu
[2]Department of Mathematics
{abeer.hasan}@humboldt.edu
Humboldt State University
1 Harpst Street
Arcata, California 95519

Abstract

We describe the effects of three semesters of a newly implemented peer tutoring program at Humboldt State University, which is classified as a Hispanic-Serving Institution. The peer tutoring program narrowed the gap between Under-Represented Groups (URGs), Pell Grant recipients, Females and First-Generation students versus the overall student population. Statistical methods were used to test whether tutoring has helped to reduce this gap. Our results suggest that tutoring not only shrunk the achievement gap but it reduced the failure rate over 50%.

1 Introduction

Student success is a widely studied problem in computer science education. Much of that research has focused on the overall student success rate while ignoring traditionally Under-Represented Groups (URGs). At our institution, California State University, women and minorities remain under-represented groups whose success requires urgent attention.

*Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Our University is one of 23 campuses in the California State University (CSU) system. Unlike the research-focused University of California system, the CSU focus is on teaching and educational access. Most campuses in the CSU have only a handful of master's programs, hence they are typically classified as Carnegie IIA institutions [4]. The CSU comprises almost half a million students and is one of the most diverse student bodies in the United States [8]. Our University in particular is classified as a Hispanic-Serving Institution, and first-generation students also are well-represented in our classrooms.

Like many institutions, our computer science program has gateway courses which impede progress of URG students. Our MATH 253: Discrete Mathematics course has traditionally been such a gateway course. In an effort to address this, we decided to start a peer tutoring program not only for this course but for all our bottle-neck and gateway courses. Peer tutoring is known to help increase the student retention rate [6] as well as the overall passing grade [1].

As reported by Hart [2] peer tutoring increases student motivation towards learning. But merely making tutoring available does not necessarily guarantee success [7]. Success also requires mentorship for the tutors, dedication from the students requesting assistance, and possibly adjustments to the tutoring format. We hypothesized that a more carefully structured implementation of peer tutoring could help close the achievement gap.

Figure 1 describes the distribution of URG students in the CS program over the past five academic years. We see a growing proportion of URG students over time. As our student population becomes more diverse there is greater need for programs such as peer tutoring to help close the achievement gap. Here we report on a comparison of student performance across three sections of Discrete Mathematics offered by different faculty with a combined total of 87 students. We aim to answer the following question: "Can peer tutoring help reduce the achievement gap between URGs and non-URG students?"

2 Methodology

Our peer tutoring program was the first offered for Discrete Mathematics and the other courses it supported. Tutoring services were offered on a first-come first-served basis for Discrete Mathematics. The experiment ran for three consecutive semesters (Spring 2018, Fall 2018 and Spring 2019). The three courses were taught by three different instructors and the courses combined had 87 students. Students had the option of attending free peer tutoring services when they needed help and were encouraged to do so by all three instructors. Tutoring was offered Monday through Thursday in the evening for a total of 20 hours per week. Tutoring attendance data were tracked. Only 19 students chose to take advantage of the tutoring services. There is no indication that only weak

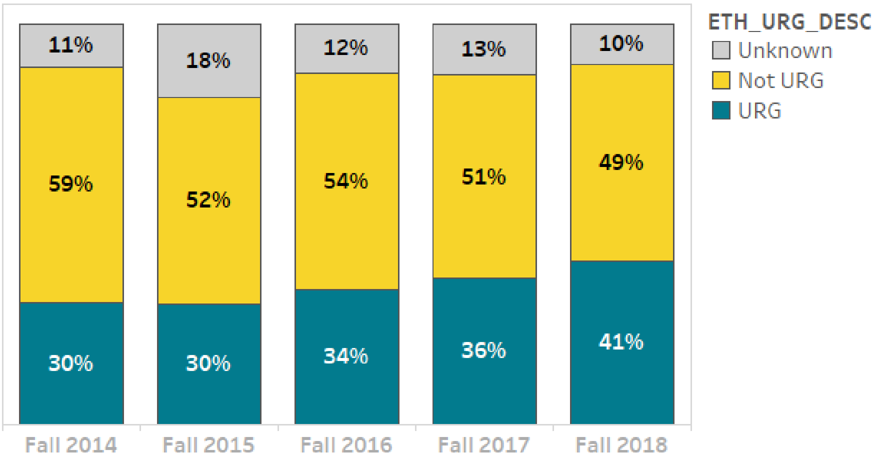


Figure 1: Student Enrollments

students participated in the program, but rather a mixture of students with different experiences, backgrounds and skills. The CS program has on average 180 students.

The tutor helps only one student at a time and restricts the contact to no more than 5 minutes when the center is busy. The tutor is instructed how to guide students to solutions through the questions and similar examples without solving the homework for them. Tutors do not just review homework, but help students develop confidence in reaching their answer and becoming independent learners. On average each tutor is trained 16 hours over a period of 2 to 3 days. This is a paid mandatory training. The training was conducted by experts at our Center of Teaching and Learning in collaboration with the university Learning Center. Currently the tutors are expected and scheduled to be able to help with more than 3 courses, including all of the 100-200 level CS department courses. All of the tutors are selected after completing our gateway courses (CS 2 and Discrete Mathematics) which are often taught by the peer-tutoring program creator. Prospective tutors must earn an A grade on their overall homework and at least an A- on their course grades. Those who display good communication skills are invited to apply for the position after a brief interview. We started with 4 tutors the first semesters (All males, 1 URG) and now we have 8 (5 males, 3 females) with a combined 4 LatinX students (URG) and one female non URG.

Since instructor approaches could account for substantial variance in success rates, all three instructors agreed to teach using the same text, *Discrete*

Mathematics with Applications by Susanna Epp, and kept materials handed out (including syllabi, assignments, and study guides for exams) as similar as reasonably possible given their teaching styles. Exams were crafted with an effort to keep problems at similar difficulty levels and content mixes (reuse of exams was impossible since failing students would reappear in the later semesters). Final grade formulae were kept similar, though differences necessarily existed due to components graded and slight percentage weighting differences. One instructor taught the Spring 2019 semester as a new preparation while the other instructors were more experienced with the course, but this was not detected in the data. Likewise, the small adjustment from the 4th Edition of Epp's text to the 5th Edition for the third semester does not seem to have affected data appreciably.

There were some noteworthy differences between the instructors' offerings. The Fall 2018 instructor used in-class worksheets to increase student engagement, but the other two did not. And the Spring 2019 instructor was unable to complete the final chapter of the outlined course owing to time lost due to both illness and inexperience with the class. We remain uncertain how these differences may have affected the data.

We used the following formula to compute the students' success rate based upon counts of final grades assigned:

$$Success\_Rate = |A, B, C| / |A, B, C, CR, D, F, NC, W, WU, I|$$

Hence a "success" in this view is that the student passed the course at a level adequate to continue on in the computer science program.

3 Results and Statistical Data Analysis

We obtained institutional data on student demographics and compared success rates for specific groups of students. Tables 1, 2, 3 and 4 summarize success rates for students who declared themselves members of Under-Represented Groups, First Generation, Legal Sex or Pell Grant versus the rest of the students. Figures 2, 3, 4, and 5 illustrate the success rates for these partitions.

We used Fisher's exact test to check if the observed differences in the success rates are statistically significant (all of the p-values are above 0.05). Table 5 shows the p-values obtained by applying Fisher's test to each of the contingency tables. We used Fisher's exact test because the sample size is small.

We conclude that after applying peer tutoring, there is no statistically significant achievement gap. It is interesting that only 19 students out of 87 attended tutoring at least one tutoring session during the semesters given that students are always asking for help. Table 6 summarizes the characteristics of those students.

Table 1: URGs

	Fail	Pass
URG	7	27
Not URG	8	32
Unknown	6	7
Total	21	66

Table 2: First Generation

	Fail	Pass
First Gen	11	35
Not First Gen	7	25
Unknown	3	6
Total	21	66

Table 3: Legal Sex

	Fail	Pass
Female	5	17
Male	16	49
Total	21	66

Table 4: Pell Grant

	Fail	Pass
Pell Grant	15	38
No Pell Grant	6	28
Total	21	66

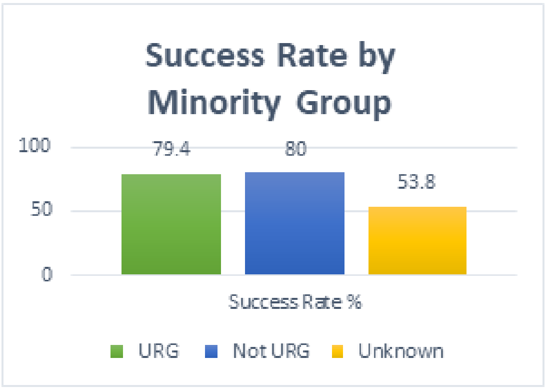


Figure 2: Success Rate by Minority Classification

The failure rate among the non-URG students is 20% (one of the lowest failure rate groups) whereas the failure rate among those who attended at least one hour of tutoring is 10% even though the majority of those who participated in tutoring were at risk students. So students who attended at least one hour of tutoring had less than half the failure rate of all other student groups.

Figure 6 was obtained from institutional data that shows the achievement gap between URGs and non-URGs before and after the peer tutoring was established in spring 2018. It is hard to interpret the graph given the fluctuation in the number of enrolled students and the fact that some students chose to not disclose their URG status. However, we note that the achievement gap

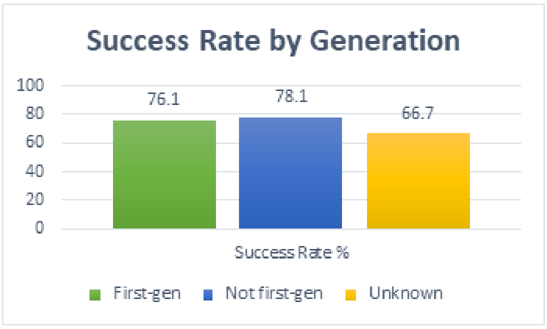


Figure 3: Success Rate by Generation Status

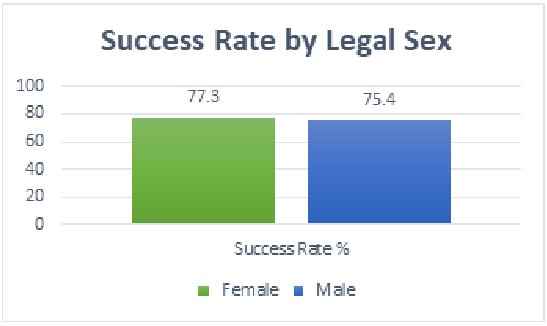


Figure 4: Success Rate by Legal Sex

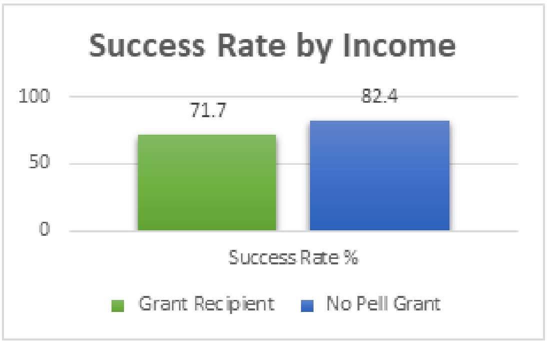


Figure 5: Success Rate by Financial Need

Table 5: Test of Statistical Significance

	P-value for Fisher's Exact Test
Under-Represented Group	0.152
First Generation	0.817
Legal Sex	1.000
Pell Grant	0.311

Table 6: Student Demographics and Tutoring Participation

	Percentage Who Participated in Tutoring
Pell Grant Recipient	90%
First Generation	74%
Females	53%
URG	47%
URG Status Not Declared	15%
All Students	22%

between URG and non-URG students is closed in the final semester. We will gather further data to see if this continues. We also note that students who choose not to disclose their URG status generally fare less well than either URG or non-URG students.

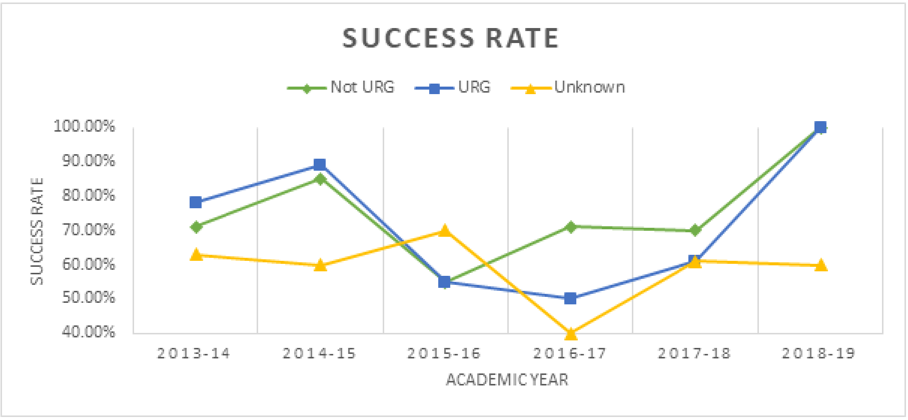


Figure 6: Achievement Gap

4 Conclusions and Future Work

We note that our peer tutoring study suffers from limitations common to research in small computer science programs. In particular:

1. The data did not come from a randomized experiment so the effect of lurking factors could not be accounted for.
2. The courses were taught by three different instructors who used slightly different grading weights, so the effect of the instructor and the grading weights could not be separated from the effect of tutoring.
3. The students themselves differed between offerings. In particular, we have no control over or adequate student representation for ethnic or cultural differences, which are known to affect success in STEM disciplines [3].
4. The fact that tutoring was available does not imply that every student utilized it. Students who chose to seek help benefited from this service while others did not.
5. Only 22% of the students enrolled in the three sections of this class participated in tutoring program. Instructors should promote tutoring and encourage students to utilize it to maximize the benefit.

Despite this, we are confident we have demonstrated that peer tutoring may have a positive impact on success rates of URGs. But why might this be so?

Our approach to peer tutoring may have created a sense of normalcy for URGs who have fewer college graduates in their social networks. Faculty encouragement may have led some to attend sessions and not feel awkward doing so. Some may have persisted because of the one-on-one environment, where dominance or competition with other students no longer existed. A success with this likely would increase self-efficacy. Simon et al demonstrated through motivational modeling that this could contribute to student success in a quantifiable manner [5]. It is worth noting that Figure 6 showed students who don't declare URG status are particularly susceptible to failure. We hypothesize that these students in particular lack self-efficacy and that interventions targeting this may help.

Hence, we are considering further data gathering and analysis of our results in light of this research to see if motivational models predict success. If so, we believe it may be possible, through short questionnaires, to identify students likely to fail Discrete Mathematics at the start of the course, and redirect them through peer tutoring and other compensatory pedagogy to improve their success rates.

References

[1] Rolando Garcia, Juan C. Morales, and Gloribel Rivera. The use of peer tutoring to improve the passing rates in mathematics placement exams of engineering students: A success story. *American Journal of Engineering Education*, 5(2):61–72, December 2014.

[2] Gail Hart. Peer learning and support via audio-teleconferencing in continuing education for nurses. *Distance Education*, 11(2):308–319, 1990.

[3] Gary Huang, Nebiyu Taddese, and Elizabeth Walter. *Entry and persistence of women and minorities in college science and engineering education (NCES 2000-601)*. National Center for Education Statistics, Washington, DC: U.S. Department of Education, 2000. Project Officer, Samuel S. Peng.

[4] American Association of University Professors. The annual report on the economic status of the profession, 2017-18. *Academe*, 104(2), March-April 2018.

[5] Rebecca Simon, Marc Aulls, Helena Dedic, Kyle Hubbard, and Nathan Hall. Exploring student persistence in stem programs: A motivational model. *Canadian Journal of Education*, 38(1):1–27, 2015.

[6] Vincent Tinto. Research and practice of student retention. *Journal of College Student Retention*, 8(1):1–19, 2006-2007.

[7] K. J. Topping. The effectiveness of peer tutoring in further and higher education: A typology and review of the literature. *Higher Education*, 32(3):321–345, October 1996.

[8] The California State University. The california state university 2019 fact book. `https://www2.calstate.edu/csu-system/about-the-csu/facts-about-the-csu/Documents/facts2019.pdf`. Retrieved 6/7/2019.

MICE: A Holistic Scorekeeping Mechanism for Cybersecurity Wargames[*]

Tristan Saldanha, Quinn Vinlove, Jens Mache
Lewis & Clark College, Portland, OR 97219
{tristansaldanha, quinnvinlove, jmache}@lclark.edu

Abstract

Cybersecurity wargames are some of the best tools for teaching security skills to groups of students, but the computational complexity of these games has increased disproportionately with the ability to measure the progress of the game. This paper introduces "Mice", a new way of assessing security skills such as detecting malware, network intrusion, and network defense, which will allow for complex games to be scored and tracked in a way that traditional score keeping can not. Mice are adaptable to any kind of simulation and are easy to use for students and educators, promising more effective learning from a wide range of security exercises.

1 Introduction

Network Wargames are highly complex tools for teaching offensive and defensive cybersecurity. In Wargame-like simulations, security students are placed together on a network and told to attack other hosts or defend from attacks from other players. However, the complexity of these simulations is often paired with a scoring system that can't keep up with the nuances of the game as it evolves. Players can tell when this mismatch occurs, and the result is that players will play the game at the level of the scoreboard, rather than at the higher level of the environment. This can leave overhead in the form of aspects of the virtualized environment that are not utilized by players; these unutilized parts of the environment resulting in the lost benefit to students and wasted effort

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

(and money) of educators. Limited grading schemes for wargames can defeat the purpose of simulating a real network. simple point markers like flags can be focused in on, allowing students to disregard the real system administration work of seeking out security problems as they appear. Wargames need a better scoring system that allows for a more realistic simulation that doesn't give students a short list of places to look for hackers, but rather encourages them to identify the signs of breach and work backwards to find malware. The system introduced in this paper, called Mice scorekeeping, is a new way of evaluating these kinds of wargames. The Mice solution is adaptable to any imaginable type of wargame; it can be used alongside other scorekeeping systems of an administrator's choosing, and can measure aspects of network control in a way that is fundamentally unique and currently unmeasurable by the flag or uptime based systems in use today. Mice, short for Miner-Imitating Counting Executables, are small programs that will add to a player's score for their time kept running and the amount of computing power they are allocated. A given singular "Mouse" will search for solved hashes, in the style of a Hashcash proof-of-work system [1], similar to how cryptocurrency mining works. Upon finding a solved hash, the Mouse will send proof of this solution to the scoreboard, thus incrementing a player's score proportionally to the approximate computing time needed to solve an average hash.

2 Capture the Flag Competitions and Current Scoring Mechanisms

CTFs, or Capture The Flag competitions, are potentially powerful vehicles for cybersecurity training [2, 4, 6, 3]. CTFs are usually built on task-specific challenges. Example problems in a CTF may be to reverse-engineer a password verification program, or to decrypt a specific conversation encrypted with an unknown algorithm. These simple tasks can be reasonably proven completed with a simple flag or text file at the end of the process- if a user can submit a plaintext string from an encrypted communication, this demonstrates that the student must have decrypted the conversation. Task-specific challenges like this can be easily proven completed because the task itself is straightforward; there are a small number of ways to complete the challenge. As more training scenarios have evolved to become more complex, assessing the state of the game has become more challenging . The CTF style flag-based proof of solution system does not translate well to a large scale live environment like a virtualized network with many hosts.

3 The Evolution of Live Testing Environments and Problem of Assessing Complex Security

Thanks to advancements in cloud computing, entire networks of computers can be virtualized to raise the bar of a capture the flag like exercise [5]. Instead of competition challenges being limited to working with small files like packet captures or individual programs, whole applications, hosts, and group of hosts can be virtualized and accessed over a web interface to engage students in more complex simulations and teach deeper skills. Examples of this new higher capacity simulation include websites like Hackthebox [9], EDURange [11], and OverTheWire [10], which let students test their penetration testing and system administration skills in a way that would be impossible in a file-specific exercise like classic capture the flags. These modern environments are more complex to evaluate than their simpler predecessors. Assessing the security of a host machine or the degree of intrusion and stealth on a network is difficult due to the variety of possible ways to engage with a host. A check that relies on looking for specific malware will miss newly discovered exploits, and measuring system uptime would disregard attackers who have exfiltrated data and locked in their own remote access, to name just a few possibilities. There is simply no one-size-fits-all solution to measuring the overall integrity of a network under attack. A grading system that lists specific criteria for a defending team will always result in a set of priorities being established and, by extension, designate a wide swath of network activity that can be ignored until something goes grossly wrong. This is presenting a misinformed view of security to students, who should instead be taught to seek out intrusion before the signs of it are obvious. Redesigning the scoring system for network wargames will result in better, more valuable exercises for students to learn from. The CCDC Competition [2] is a well known blue teaming exercise. The game is scored using primarily service uptime as the metric of a well-secured network, along with injects and writing incident reports to a lesser extent. Keeping services up and running is an aspect of network security, but this grading scheme identifies a list of services that allow blue teams to focus on a handful of tasks, while leaving the scoring system blind to whether an attacker has gained access to the network or not. A common scenario that occurs in these kinds of games is that the attacking team will take some time to establish their persistence and elevate privileges in host machines, before attacking scored objectives. This phase of the attack, when nothing has visibly gone wrong to the untrained sysadmin's eye, is being ignored by modern score systems, which is a disservice to both teams playing the game. Mice can fill this gap. Mice are less ambiguous in terms of detection - simple process monitoring commands will show the majority of mice running on a system. This limits the amount of stealth that the red team can use, and gives blue teams more to look for. Speeding up the silent intrusion and elevation of privilege phase of the red team's gameplay improves the experience for both teams.

4 The MICE System

We present a new way of evaluating network security during live testing environments, using what we call Mice (henceforth referred to as a single Mouse program or several Mice programs, for readability). Students will be given small python programs that will add points to their score while running, called Mice. These Mice can be run on any machine on the given exercise network to add points to a player or team's score. The Mouse program is undergoing continuous active development [7], to keep the software as simple as possible while including all necessary functionality to be used effectively. The two variables that matter to a Mouse include the Mouse's ownership and it's speed. Each Mouse has an "owner", which is the player or team who receives the points that the Mouse generates while running. A player would want as many mice as possible to be running under their own ownership. The "Speed" of a Mouse is the rate at which it gains points. The more system resources a Mouse is given, the faster it will gain points. This incentivises players to be mindful of how hard they push their Mice; an unrestrained Mouse may utilize too many system resources and cause a user's machine to slow down or even crash. A hostile Mouse placed on an opponent's machine would have the same effect, which could result in detection of foreign Mice. This adds a level of complexity to a player's Mouse strategy; a quiet, undemanding Mouse may avoid detection and garner points slowly, while a Mouse taking lots of system resources to gain points fast may lead to its detection and removal.

5 Blockchain-inspired Mouse Design for Realism in Threat Modeling

While the Mouse scoreboard system itself does not maintain a complete blockchain, much of its design is owed to the proof-of-work mining process that makes Blockchains secure, or, colloquially, "mining". Mice measure their speed and contribution to player's score by computing hashes, such that the main objective is to keep as many of your Mice running as possible with as many system resources obtainable so as to mine the most hashes. The "points" that mice generate are measured in hashes. While a mouse is running, the work it does is search for a proof-of-work "solved" hash- that is, a hash of a random string that ends in an arbitrary number of zeroes. In practice, these solved states are called "solves" or "solved hashes". Since each character in a SHA256 hash has a 1/16 chance of being any character, the probability of generating a hash ending in n many zeros is $(1/16)^n$. For a difficulty 5 hash, the MICE default, this is approximately one per million hashes, or one per megahash. When any mouse finds one of these solved hashes, it will send this solution to the scoreboard, and that mouse's team will be awarded one megahash of points. The blockchain style of the Mouse program has several benefits:

5.1 Realistic Threat Landscape

With the rising value of cryptocurrencies in the last few years, "Cryptojacking" is a very real threat to modern networks [8]. So-called Cryptojacking attacks involve attackers placing mining malware on remote servers to steal computing power for financial gain. Identifying and removing this kind of cryptocurrency mining malware is a real task that modern system administrators have to deal with. Building a network defense simulation around finding and destroying unwanted Mice on a host is a relevant primer for the current threat landscape experienced by network security professionals.

5.2 More Ways to Play

Using more complex programs as objects of the game, as opposed to finding text files on one another's machines, allows students to play their games in more nuanced ways. For example, rather than simply deleting a hostile Mouse found on one's machine, why not change its owner and make it gain points for your own team? Or better yet, modify a well performing player's Mice to belong to you, and let other players infect one another with your mice to build your own personal botnet. Using these programs in a more complex game allows players to utilize elements of reverse engineering, and grapple with the overlap between offensive and defensive security skills.

5.3 Simple Code

A priority during Mouse development is to keep the code simple enough that users can feasibly make modifications to the code during an ongoing exercise. The Mouse program is written in Python, which can be edited without needing to be recompiled, and is a high enough level language that modifying simple variables and methods can be done without necessitating an unreasonably high level of programming expertise. At the time of writing, the open-source Mouse program is receiving regular updates and improvements on its GitHub page, linked below.

5.4 Compatibility with Other Scorekeeping Mechanisms

The Mouse system is not perfect for some teaching objectives. The way that Mice go about incrementing points for a user is highly dependent on the hardware of the host they run on. In reality, the true value of an arbitrary machine may not directly correlate with its hash-solving potential. This is why the Mice system is made able to run alongside other, more traditional flag or service-uptime based systems. Nothing about the Mouse system limits educators from layering Mice on top of other systems that may better reflect the specific goals of a lesson- for a course in Windows Administration, a service-uptime scorekeeping style may be more effective than in a general pentesting simulation, while a Mouse system running on the side can also be in place to subtract student's points based on discovered remote code execution vulnerabilities. The ultimate purpose of the Mice is to aid educators in creating better simulations; to that end, flexibility and adaptability have been the main priorities through the development and design of the system.

6 Different Uses of Mice Scorekeeping

The rules for how to use the Mice can change to best suit the kind of war game being played. Mice are lightweight, simple, and their usage can be adapted based on the nature of the simulation they are applied to. Here, we will outline some ways that the Mice scoring system can be put to use in many different types of network security simulations.

6.1 Player versus Environment Simulation

For a player vs. environment simulation, like a firewall exercise, players may have Mice that they have to keep running against an incoming, possibly automated attack. In this case, a user would have only their own Mouse or Mice under their own ownership, and would have to keep their Mouse safe and running while defending from an incoming intruder looking to disable their Mice. This type of game may teach skills on the more defensive side of security, like obfustication of critical processes and host hardening. A player would also need to be able to detect intrusion as fast as possible, since any time with their mouse not running would cost them points.

6.2 Blue Team Versus Red Team Simulation

In a blue team / red team simulation, a blue team tries to keep the red team's score as low as possible while the red team tries to run their Mice on blue team machines. There may be only a single team of Red Mice, or for a more complex game, a Blue and Red team of Mice. The blue team may be responsible for finding and stopping as many Mice as they can while having no Mice of their own to run, while the red team can only run Mice on the blue team's network. Or, if the simulation is desired to be more complex, there may exist both blue and red mice on the network, while Mice are only allowed to run on blue team hosts, thus making the players of both teams search for enemy mice on the limited number of hosts and disable them. This teaches the red team network intrusion and, more uniquely, stealth tactics, as the red team needs their Mice to go undetected for as long as they are able. This game would also be providing a valuable exercise for the blue team to practice identifying the signs of malware running on their workstations, with time being a factor in their response abilities.

6.3 Player versus Player Simulation

In a player vs. player scenario, players may be allowed to sabotage the other player's Mice and attempt to install their own Mice on other machines. This kind of simulation allows for the greatest range of creativity for students. This exercise teaches a huge range of skills, including workstation hardening, penetration testing, intrusion detection, reverse engineering, and more. It teaches the balance of offensive and defensive tactics in a way that is utterly unique to other forms of scoring, and allows a live network security simulation unparalleled by artificial tools and limited evaluation metrics.

6.4 More Possible Use Cases and Abstractions

More ways to use the Mouse system are discovered as the idea is shared with more educators with different experiences and expertise. Below are some ideas of ways to put the Mice system to work in a variety of more abstract types of simulations.

6.4.1 Firewalls

Imagine a networking game being played on a network with many hosts. Each host could have a mouse server running on it, but each behind different firewalls configured in different ways; an exercise could be made of "unlocking" these firewalls so that these scoreboard could be reached by Mice, with points representing uptime that the scoreboard was reachable.

6.4.2 Network Mapping / PING Sweeping

Another possible exercise could place scoreboards on many different ports of a given machine, encouraging students to learn to use tools like Nmap to discover hidden scoreboards or mice. The score at the end of the game could in some way reflect the amount of mice that were linked to the right scoreboards; perhaps some mice would be best fit to certain boards, being locked to broadcast on a specific port within the python script. The possible implementations to the Mice system are truly limitless. The simple code makes the programs adaptable, and the uses of the hash-based scorekeeping are the perfect way of measuring time, persistence, and compute resources managed by a single player.

7 Classroom Tests

To simulate what a real world applications might look like, a group of research assistants played a sample game with the Mice programs. We played a free-for-all style of game; each of us had our own client machines on the same subnet. We played with 4 players, each of us given a machine on the same subnet. A scoreboard was set up on a fifth machine, and we agreed to leave it alone from tampering.

7.1 Software Sabotage

7.1.1 Hardcoded Variables

The Mouse program sets its team ownership through a command line argument. If this argument is altered, the mouse runs for a different team. One player modified an opponents mouse program to immediately discard this argument variable, and hardcoded their own team name inside the program.

7.1.2 Slowed Mining

Other software changes were made to other teams to damage their mice. One was slowed down by adding some extra work to the hash searching function. In this case, the variable “fish” was created, assigned the value “glub glub”, and then discarded every time a hash was checked. This didn’t affect the program in any other way than to roughly double the hash searching loop, slowing down opponent’s point scoring.

7.1.3 Tampered Debug Outputs

Another change removed the debug outputs from one mouse program, requiring the student to spend time downloading a fresh copy of the Mouse program to understand what was going wrong with it as it was failing to run. One student went a step further by locking the debug outputs to show that a mouse was running correctly, while all it was actually doing was printing the expected outputs and submitting no points.

7.2 Propogating Faulty Mice

This is the behavior that was most anticipated in the early phases of designing this mouse system. All four students attempted in some way to run their mice using the hosts of other machines. On the lab machines this test run was carried out on, all players knew all other user's credentials (username and password student), and both SSH and FTP were enabled on all machines. Some students copied their mice to other machines, while others just ran copies of the other player's mice under their own name.

7.3 Player Secured Mice

Given the above examples of players tampering with each other's mice, one player bypassed this risk by modifying his mouse and compiling it to a .pyc file before using it. The player hardcoded his own team name into that mouse and made sure that it was working properly, then played the game with his compiled mouse rather than the stock python file. Using the compiled version hardened his mice from attack, and he was able to use them without having to check if they had been modified while he wasn't watching.

8 Codebase

The code for the Mice system is being developed and is openly available in this github repository: https://github.com/kh3dron/mice-scoreboard. The code is still undergoing development at the time of this writing, but mainly feature updates; the current version of the system is ready to be put to use. Two Python scripts are involved in the Mice: the Mouse client, called mouse.py, and the scoreboard, called server.py. The client is a short program, at the time of writing only about 60 lines long. The program will search for a proof of work hash solution until one is found, at which point it will send the solve to the server, wait for positive confirmation of reception, then go back to searching for a new solve. The simplicity of the Mouse client introduces some fairly large security holes, which either team can exploit to their own advantage, or possibly repair to make the Mouse run more securely. While the Mouse development is ongoing, the goal is not to make the program immune from abuse. The simple and imperfect nature of the Mice introduces yet another nuance to their usage, encouraging a baseline of scripting knowledge as a way to give players an extra edge on one another during an exercise.

9 Upcoming Features

The Mouse and Scoreboard programs are both complete enough to be used as they are, but more updates are coming primarily to improve the data from the server. Some planned updates include: server-side difficulty requirement broadcasting to manage traffic and server logs/a proper administrator dashboard. A more detailed account of upcoming software additions can be found at the GitHub repository for the project, as linked above.

10 Acknowledgements

This material is based upon work supported by the National Science Foundation under grant numbers 1723705, 1723714, 1516100 and 1516730. We would like to thank Lewis & Clark students Alex Lotero and Kris Gado, as well as professor Richard Weiss from the Evergreen State College.

References

[1] Adam Back. Hashcash - a denial of service counter-measure. `http://www.hashcash.org/papers/hashcash.pdf`.

[2] CCDC. CCDC competition rules and guidelines, 2019. `https://www.nationalccdc.org/index.php/competition/competitors/rules`.

[3] Andy Davis, Tim Leek, Michael Zhivich, Kyle Gwinnup, and William Leonard. The fun and future of CTF. In *2014USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[4] Patrick Hulin, Andy Davis, Rahul Sridhar, Andrew Fasano, Cody Gallagher, Aaron Sedlacek, Tim Leek, and Brendan Dolan-Gavitt. AutoCTF: Creating diverse pwnables via automated bug injection, 2017.

[5] George Louthan, Warren Roberts, Matthew Butler, and John Hale. The Blunderdome: An offensive exercise for building network, systems, and web security awareness, 2010.

[6] Cheung R. S., Cohen J. P., Lo H. Z., Elia F., and Carrillo-Marquez V. Effectiveness of cybersecurity competitions, 2012.

[7] Tristan Saldanha. MICE scoreboard codebase. `https://github.com/kh3dron/mice-scoreboard`.

[8] Symantec. What is Cryptojacking? how it works and how to help prevent it. `https://us.norton.com/internetsecurity-malware-what-is-cryptojacking.html`.

[9] Hack the Box. Hack the box, 2019. `https://www.hackthebox.eu/`.

[10] Over the Wire. Wargames, 2019. `https://overthewire.org/wargames`.

[11] Richard S. Weiss, Stefan Boesen, James F. Sullivan, Michael E. Locasto, Jens Mache, and Erik Nilsen. Teaching cybersecurity analysis skills in the cloud. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE 15*, 2015.

Mining GitHub Classroom Commit Behavior in Elective and Introductory Computer Science Courses*

Gina Sprint and Jason Conci
Department of Computer Science
Gonzaga University
Spokane, WA 99258
sprint@gonzaga.edu, jconci@zagmail.gonzaga.edu

Abstract

Git and GitHub are the industry standard tools used for version control today. Consequently, integrating Git and GitHub into CS curriculums is of increasing importance for educators. With GitHub Classroom, students can submit their programming assignments via GitHub instead of via a traditional learning management system. In this paper, we present our experience using GitHub Classroom in two elective courses and in a CS1 course. We also present our results from mining student commit behavior from programming assignments submitted via GitHub Classroom in these courses. While we found a weak correlation between commit behavior and grades at the group level, we did find individual students exhibited strong correlations. Furthermore, our data analysis indicated that students responded to a small amount of graded points being allocated for quantity and quality of commit messages by improving their use of version control. Lastly, based on student responses, we conclude that introducing version control in CS1 is not too early and we encourage educators to consider adopting GitHub Classroom for assignment submission as early in the curriculum as possible.

*Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Version control systems are industry standard tools used for several purposes, including tracking file changes, for team collaboration, and for bug/issue management. Consequently, educators have long been teaching version control tools like Git, CVS, Subversion, and Mercurial in upper-division computer science and software engineering courses. Students typically learn these tools in a software engineering course during their sophomore or junior year, then apply these tools and skills on a team-based senior design project. Because of the importance of version control for industry, educators have recognized a need to push version control earlier in the curriculum, some as soon as introductory computer science courses (CS1) [5, 1].

Recently, the most popular version control system being taught in higher education is Git [1]. GitHub is a web-based platform for managing code projects under Git version control. When used in computer science education, GitHub has been shown to predict better learning outcomes and classroom experiences [4]. In fact, for computer science courses, GitHub is often a preferred alternative to a learning management system (LMS) for hosting instructor example code and for student programming assignment (PA) submission, collaboration on group projects, and for receiving feedback on work. In 2015, GitHub recognized the need for educators to use GitHub as an LMS alternative when they released GitHub Classroom, a Git and GitHub-based assignment management and submission system. To submit assignments via GitHub Classroom, students are required to learn the basic stages of Git version control usage, including creating a repository, adding files to the staging area, committing files with messages, and pushing files to a remote repository (e.g. GitHub). In this paper, we present results from mining student use of GitHub Classroom for programming assignment collection in three different courses. We also present our results of implementing an intervention to motivate students to commit code changes with higher quantity and quality. Finally, we present commit behavior and perception of GitHub Classroom from CS1 students who were exposed to Git, GitHub, and GitHub Classroom for the first time.

2 Related work

Researchers have focused on using version control systems in educational settings to monitor and/or mine student use of version control systems [5], [4, 2, 6, 8, 9], predict student grade performance using machine learning techniques [8], [3], motivate students to commit more via gamification [10], characterize teamwork [9, 7], and investigate the relationship between commits for coding and testing [6]. Common features extracted from version control repositories include number of commits, size of commits (file size, number of files modified, or quantity of insertions/deletions), lines of code added, and timing of commits (hour of day of commit, spacing of commits, commit streaks of committing regularly for several days, or relationship to a deadline and/or milestone).

Around 2004, literature emerged that investigated using version control systems to monitor and/or mine student programming behavior. Liu and colleagues analyzed student CVS repositories to investigate student interaction in teams [7]. Mierle and colleagues correlated grades with CVS repository features, such as number of revisions per file, number of operations, time of submission, etc [8]. Using machine learning techniques, they concluded that such features could not be used to accurately predict grades. Glassy used Subversion commit logs to investigate when students worked on assignments relative to a deadline, finding that while students did perform "mega-commits" of changes, students did not tend to do a single "mass-commit" of changes for their submission [2]. Notable results related to our work include number of commits was not correlated with grade [8], students with vague commit messages generally had lower grades [2], number of code lines was strongly correlated with higher grades [8], and students increased their commits over time (and start committing earlier) as they became more comfortable with version control and appreciated its value [5].

3 Methods

In this paper, we analyzed student use of GitHub Classroom to answer the following three research questions:

1. What are the relationships between commit features and grades (overall and PA)?
2. Can we motivate students to commit code with higher quantity and quality?
3. What are CS1 students' initial perceptions of Git and GitHub Classroom?

We utilized GitHub Classroom for programming assignment collection in three courses, all of which were taught by the same instructor: an iOS app development elective (Swift, Fall 2018, N=34), a data mining (DM) elective (Python, Spring 2019, N=31), and a CS1 course (C++, Spring 2019, N=27). Thirteen of the students enrolled in the iOS course were also enrolled in the DM course. We refer to this group of students as iOS/DM (iOS DM intersection). The prerequisite for iOS and Data Mining was CS2 (C++), though the majority of each of theses courses were seniors (27/34 and 23/31, respectively). We formally teach Git/GitHub starting in the student's sophomore year, after they have solidified their command line and programming skills in their CS1 and CS2 courses. Therefore, the majority of students in these two elective courses did have prior exposure to Git/GitHub, but this was not the case for all students. In order to prepare students to use GitHub Classroom, the instructor provided a crash course on Git/GitHub in these courses. In the crash course, the motivation and intuition of version control technology was covered, then the following Git commands (with relevant options/switches) were introduced via a hands-on lab: init, add, commit, push, pull, status, log, branch, and config. In iOS, students were also taught how to use the Xcode Git/GitHub user interface tools.

While there were no prerequisite courses for CS1, our CS1 served as a CS0/CS1 hybrid for both CS and engineering majors (14/27) and non-majors (13/27). In this course, we introduced algorithmic problem-solving techniques using C++ in a virtual Linux environment. From the beginning of the semester, students were learning basic command line skills, which enabled the instructor to briefly introduce the necessary skills for students to use GitHub Classroom to submit their final PA (PA9) at the end of the semester. Command line skills that were covered in CS1 included working with paths, file system navigation, file/directory creation/removal/renaming, launching text editors, compiling source code using g++, and executing programs. For the CS1 crash course, we covered a subset of the aforementioned Git/GitHub crash course in one 50-minute class. Specifically, we covered limited information about the motivation for using version control, basic Git commands to push to GitHub, and one hands-on "Hello World" GitHub repository exercise. The students were also provided detailed instructions on how to set up a GitHub Classroom repository. Four bonus points (of 100 points for PA9) were offered to students who submitted PA9 via GitHub Classroom instead of via the LMS.

From GitHub repository log data, we summarized student assignment repository commit information using quantity (number of commits) and quality (average number of changes per commit and average number of words per commit message). For these features, we then computed the average, coefficient of variation (CV), and Pearson correlation with grades for each assignment repository. The coefficient of variation was computed by dividing the standard deviation by the mean and multiplying by 100. For our second research question, we introduced an intervention in the DM course to improve students' use of commit messages. There were eight PAs in this course: PA1 and PA2 allocated 0 graded points for quantity and quality of commit messages, PA3 and PA4 allocated 2 points, PA5 and PA6 allocated 4 points, and PA8 allocated 0 points (PA7 was a bonus PA and we omitted it from analysis). For our last research question, we asked CS1 students to fill out a survey about their experience learning Git and GitHub in the crash course and if applicable, using GitHub Classroom to submit their final PA.

4 Results and Discussion

In total, we analyzed 3,922 commit messages from GitHub Classroom repository logs. For all groups, we include the average, CV, correlation with PA grade (r_{PA}), and correlation with overall course grade (r_{course}) for each feature in Table 1. To provide more details about the commit behavior in Table 1, Figure 1 shows the average number of commits as a function of PA for the iOS/DM group.

Table 1: Summary of feature results in the form: mean (CV). * denotes correlations (r) at $p<0.05$.

Group name	Number of commits per assignment	Average changes per commit	Average words per commit message
iOS (1,845 commits)	9.04 (90.55%) r_{PA}=0.10 r_{course}= 0.80	578.37 (306.83%) r_{PA}=0.06 r_{course}=-0.05	5.07 (70.82%) r_{PA}= 0.12 r_{course}= -0.20
DM (2,030 commits)	9.35 (77.41%) r_{PA}=0.31 r_{course}=0.36*	995.48 (136.58%) r_{PA}=0.04 r_{course}= -0.29	4.74 (76.01%) r_{PA}=0.16* r_{course}=0.01
iOS/DM (1,837 commits)	10.87 (74.57%) r_{PA}=0.21* r_{course}=N/A	811.88 (241.56%) r_{PA}=0.08 r_{course}= N/A	5.11 (68.91%) r_{PA}=0.18 r_{course}=N/A
CS1 (47 commits)	3.92 (109.55%) r_{PA}=0.26 r_{course}=0.23	292.00 (72.51%) r_{PA}=0.53 r_{course}= 0.37	3.00 (46.01%) r_{PA}=-0.04 r_{course}=0.28

Similar to prior work [8], we found weak correlations between student commit behavior and student grades for both PAs and overall course grade; however, we did find several strong relationships for individual students. For example, one iOS student had r_{PA}=0.95, r_{PA}=-0.84, and r_{PA}=0.73 for commits per assignment, changes per commit, and words per commit when correlated with PA grade. This student earned an average of 90.67% on their iOS PAs, but not all high performing students had similar grade correlations. A different iOS student with average PA grade of 91.5% had opposite correlations for the same features, with r_{PA}=-0.80, r_{PA}=0.81, and r_{PA}=-0.47. This suggests further analysis is needed to categorize types of GitHub user commit styles. As expected, we observed fewer commits, changes, and message words for beginning Git users (CS1) compared to the upper division elective students. Lastly, all groups exhibited high feature variability and students in the iOS/DM group

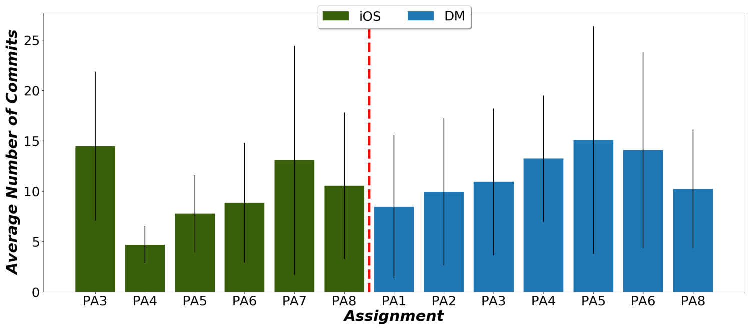


Figure 1: Figure 1. Average number of commits for each programming assignment (PA) of the iOS/DM group with iOS on the left and DM PAs on the right. Standard deviations are shown as error bars.

committed more frequently and with more descriptive commit messages than the other groups.

Table 2 shows more specific results for the DM group, specifically pertaining to the applied intervention. We include DM feature values for pre-intervention, intervention (2 points and 5 points allocated), and post-intervention. Because these PAs became increasingly long and complex over the course of the semester, we also investigated normalizing our features by dividing by the number of code lines. From pre to post-intervention, students increased their number of commits 18.42% (36.59% normalized) and increased their number of words per commit message 38.88% (64.64% normalized). Interestingly, students also increased their number of changes per commit 76.15% (278.64% normalized), despite increasing their commits. This could be due to students frequently editing their code in the larger PA and/or the differences in the PA structures. We also compared iOS/DM to non-iOS/DM students for pre-intervention PAs to measure the practice effect of an extra semester with GitHub Classroom. The iOS/DM group compared to the non-iOS/DM group had 9.19 vs. 5.33 average commits ($p<0.02$), 697.73 vs. 869.96 average changes per commit ($p=0.43$), and 3.73 vs. 3.20 average words per commit message ($p=0.31$). These results suggest that an extra semester of practice with GitHub Classroom improved student commit behavior without the need for an explicit intervention.

For our last research question, we collected qualitative feedback about CS1 student perception of version control. Out of 27 total students, 25 students attended the Git/GitHub day. Of those 25, 2 students had worked with

Table 2: Summary of feature results for each group in the form: mean (CV).

PA Set	Number of commits per assignment	Average changes per commit	Average words per commit message message
Pre-intervention	6.95 (91.29%)	797.73 (107.55%)	3.42 (60.43%)
2-point Intervention	10.07 (64.86%)	1470.62 (141.82%)	5.96 (79.74%)
5-point Intervention	11.61 (75.98%)	513.25 (98.33%)	4.84 (73.70%)
Post-intervention	8.23 (56.49%)	1405.18 (74.52%)	4.75 (48.37%)

Git/GitHub before and 12 students submitted their final PA via GitHub Classroom for bonus points on the assignment. The mean and standard deviation for the 25 CS1 student responses to the survey are provided in Table 3. The survey also prompted the student for any comment he/she would like to make regarding the crash course and/or the version control tools. For brevity, we have included the following quotes we found representative of the dataset:

1. "Always wondered how GitHub works and how programmers use it, now I have a better idea of that"

2. "Github sounds useful for coding projects!"

3. "I wish had more than one day to practice and learn about GitHub"

4. "Stress the way that git makes it possible to collaborate on code with others"

5. "I found the GitHub day valuable, but I do think we covered maybe a little too much information for 1 class period, and perhaps spreading it over 2 days would be better"

Overall, CS1 students reported that they enjoyed learning version control and all but one student clearly saw the value of Git/GitHub for software development. Interestingly, both the Likert and free response results demonstrate that CS1 students would have liked to learn Git/GitHub earlier in the semester (all students answered a 3, 4, or 5 to this question) and to have learned more, as demonstrated by student responses #3 and #4 above. We often think CS1 is too early to cover version control; however, these CS1 students challenged the tradition of waiting until a sophomore or junior-level software engineering course. It is important to note that our particular CS1 students worked with the command line all semester, which enabled us to teach command line Git/GitHub towards the end of the course. For CS1 courses that are not as command line-focused, graphical tools for version control like GitHub Desktop would be a viable alternative. Lastly, another reason to cover Git/GitHub earlier in the curriculum and in a CS1 course is to help students' knowledge become comprehensive enough that they can solve their own problems related to the tool. For example, one student commented, "The only problem I had with GitHub was attempting to upload after deleting files from the GitHub website. After a deletion happened I would get all these funky upload errors on the terminal that I couldn't figure out. But that is out of your control. Overall I did enjoy using GitHub for PA9."

Table 3: CS1 student responses to Likert questions on a scale from 1 (completely disagree) to 5 (completely agree) in the form: mean ± standard deviation.

I enjoyed learning about Git/Github for file version control.	4.04 ± 1.08.
I see the value of Git/Github for software development	4.72 ± 0.54
I would have liked Git/Github to be covered earlier in the semester.	4.12 ± 0.86
I plan to use Git/Github in the future.	3.96 ± 1.34

5 Summary and Future Work

In this paper we analyzed nearly 4,000 commit messages from GitHub Classroom repositories to investigate student commit behavior. Similar to prior work, we found weak correlations between student commit behavior and student grades at the group-level. We also found that students did respond to a small amount of graded points being allocated for quantity and quality of commit messages. Lastly, based on student responses, we concluded that introducing version control in CS1 is not too early. To the contrary, we encourage other educators to consider adopting GitHub classroom for programming assignment submission as early as CS1. For future work, we plan to explore the effects of different approaches to motivate students to commit early and often, such as gamification. We also plan to integrate version control more in our CS1 curriculum while still maintaining the success of our one day "crash course" on the subject.

References

[1] M. A. Angulo and O. Aktunc. Using GitHub as a teaching tool for programming courses. In *Proceedings of the 2018 ASEE Gulf-Southwest Sect. Annu. Conf.*, pages 1–4, 2018.

[2] L. Glassy. Using version control to observe student software development processes. *Journal of Computing Sciences in Colleges*, 21(3):99–106, 2006.

[3] Á. M. Guerrero-Higueras, N. DeCastro-García, V. Matellán, and M. Á. Conde. Predictive models of academic success: A case study with version control systems. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 306–312, 2018.

[4] C. Hsing and V. Gennarelli. Using GitHub in the classroom predicts student learning outcomes and classroom experiences: Findings from a survey of students and teachers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 672–678, 2019.

[5] J. Isacson and E. Lindblom. *Correlation of User Behaviour Patterns and Assignment Supplements in KTH-GitHub Repositories.* 2017.

[6] L. Baumstark Jr. and M. Orsega. Quantifying introductory CS students' iterative software process by mining version control system repositories. *Journal of Computing Sciences in Colleges*, 31(6):97–104, 2016.

[7] Y. Liu, E. Stroulia, K. Wong, and D. German. Using CVS historical information to understand how students develop software. In *Proceedings of the 2004 International Workshop on Mining Software Repositories*, pages 32–36, 2004.

[8] K. Mierle, K. Laven, S. T. Roweis, and G. Wilson. Mining student CVS repositories for performance indicators. *ACM SIGSOFT Software Engineering Notes*, 30:1–5, 2005.

[9] W. Poncin, A. Serebrenik, and M. van den Brand. Mining student capstone projects with FRASR and ProM. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, pages 87–96, 2011.

[10] L. Singer and K. Schneider. It was a bit of a race: Gamification of version control. In *2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques*, GAS, pages 5–8, 2012.

Teaching Math for Computer Science in an Open-Enrollment College – an Applied Learning Experience[*]

Baoqiang Yan
Computer Science, Mathematics and Physics
Missouri Western State University
Saint Joseph, MO 64507
byan@missouriwestern.edu

Abstract

Computer science majors need considerable mathematical background to understand important topics such as token recognition in scanning, cryptography, graphics and simulation among others. Regular math courses may cover relevant abstract concepts without a specific computer application context, leaving students puzzled about why they are needed and where they are applied in computer science. This issue may get even more challenging in an open-enrollment college where many students tend to be less-prepared academically. This paper describes the author's experience of employing applied learning to bridge the gap between math concepts and their applications in computer science. The selected math topics are all programmable with various tools, and students are exposed to more tangible program exercises with limited theoretical discussion.

1 Introduction

It is a common understanding that many computer science (CS) topics have a deep link to mathematics under the hood. According to ACM Computer Science Curricula 2013 [3], mathematics for CS can be divided into foundation mathematics and those that serve a specific CS area. The author believes that

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

this two different mathematics in CS should be taught differently. Furthermore, students without significant prior mathematics background should not be prohibited from pursuing a major in CS. This is especially important for a CS program in an open-enrollment college like us because many students tend to be less-prepared in mathematics.

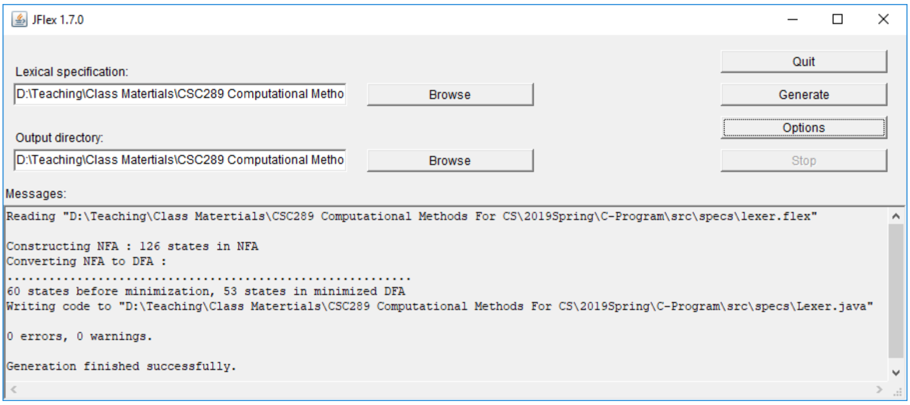
To alleviate the concern of mathematics being a hurdle in our CS program while trying to stick with the ACM CS Curriculum Guidelines for future ABET accreditation, we have incorporated in our curriculum a special mathematics course named "Computational Methods for Computer Science". Different from the traditional math courses that do not take into consideration how abstract math concepts are applied in computer science applications, this course is designed to tie all covered topics with specific computer programs and explains why they are needed and how they are applied with limited theoretical discussion. The immediate tangible outputs from these programs after the corresponding math concepts are applied really help bridge the student's understanding gap between concepts and applications. Students engage themselves in hands-on learning of math and computer science and gain a sense of accomplishment for having successfully written a computer program. It is an example of practicing the mission of applied learning that is strongly advocated by our university. After taking this course successfully, students build a solid background for other major courses that we offer such as Compiler Theory, Simulation, Graphics, Cryptography. Therefore, we view this math course as a gateway towards these major courses.

2 Course Structure

This course covers four math topics as described in the following subsections. In a 16-week semester, each topic takes about four weeks. Students are assessed via on-line quizzes, programming assignments and exams. Each topic is explained with carefully selected tools and applied in a closely relevant computer application.

2.1 Automata Theory for Token Recognition

Token recognition is the first step of compiling a computer program written in certain language and done via a program scanner. The scanner can be either written from scratch or created automatically through a scanner generator. The former method could be rather challenging for most of our junior students and not realistic to finish in a four-week period, given that they need to learn the concepts and get familiar with the tools first. Therefore, the second strategy is employed, and JFlex is chosen as the scanner generator since it uses Java, the main programming language we teach.


JFlex 1.7.0
Lexical specification:
D:\Teaching\Class Matertials\CSC289 Computational Metho
Browse
Output directory:
D:\Teaching\Class Matertials\CSC289 Computational Metho
Browse
Quit
Generate
Options
Stop
Messages:

```
Reading "D:\Teaching\Class Matertials\CSC289 Computational Methods For CS\2019Spring\C-Program\src\specs\lexer.flex"

Constructing NFA : 126 states in NFA
Converting NFA to DFA :
.......................................................
60 states before minimization, 53 states in minimized DFA
Writing code to "D:\Teaching\Class Matertials\CSC289 Computational Methods For CS\2019Spring\C-Program\src\specs\Lexer.java"

0 errors, 0 warnings.

Generation finished successfully.
```

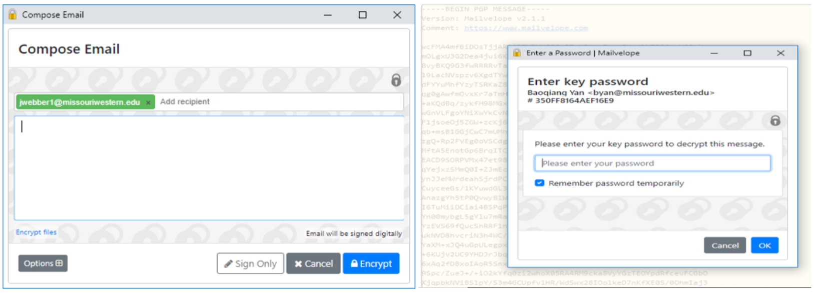
Figure 1: GUI of JFlex generating a scanner for C Minus

As shown in figure 1, the working mechanism of JFlex is briefly revealed in the bottom messages window. Automata play a major role in scanner generation. The input to JFlex is a .flex file that contains the lexical specification for the tokens allowed in the language. The output is a Java class that serves as the scanner. Besides explaining what regular expressions are and how they are represented as macros in the .flex specification file, there are additional steps in the scanner generation process that need to be demystified to the students, including how to construct Non-Deterministic Finite Automata (NFA) from regular expressions, why and how NFA should be converted to Deterministic Finite Automata (DFA), and how to further minimize the states in that DFA before it is used for table-driven implementation of the scanner. Each of these key points is discussed with at least two general examples in class. Then students are asked to apply what they learn to create a scanner for a simple language called C Minus [5] or C–. It is a stripped subset of C language with a very simple set of allowed token patterns, making it an ideal candidate to use in teaching automata for token recognition.

2.2 Elementary Number Theory for RSA

Cryptography is a critical topic in computer science and different ciphers may involve different mathematics [6]. We choose RSA as the cipher to discuss because of two reasons. First, RSA is a strong and popular public-key cryptosystem most students have heard of. Second, its algorithm can be broken down into pieces that can be described elegantly using elementary number theory.

Typical discussion on a cipher includes key generation/scheduling, encryption/decryption, why it works, computing efficiency and security level. As for RSA's key generation, students need to understand basic modular arithmetic first. Calculation of inverse, especially multiplicative inverse, plays a crucial role. Students need to know when a number has a multiplicative inverse and if so, how to compute it efficiently. For the first issue, students need to understand what coprime means, how to check if two numbers are coprime through Greatest Common Divisor (GCD) algorithm and why we typically pick prime numbers as the integer ring size although that is not required in RSA. For the second question, students need to understand that Extended GCD not only checks if the public key e has an inverse that serves as the private key but also what it is if there is one. Extended GCD is efficient than relying on the multiplication modular table for inverse search which is not realistic for big numbers. RSA uses exponentiation modular for encryption with public key and decryption with private key. Fermat's Little Theorem, closely related to exponentiation modular, helps students understand why RSA algorithm works and can simplify exponentiation modular calculation with large exponents. To understand why RSA is secure, students need to understand that the factorization of an integer is a process of finding a set of prime numbers whose product gives back the original number. Though small numbers can be used for demonstration purpose, it should be pointed out to the students that factorizing extremely large numbers is not possible given current computing technology. This helps students understand why RSA requires two large prime numbers in the first place.



Compose Email
Compose Email
jwebber1@missouriwestern.edu
Add recipient
Encrypt files
Email will be signed digitally
Options
Sign Only
Cancel
Encrypt
Enter a Password | Mailvelope
Enter key password
Please enter your password
Remember password temporarily
Cancel
OK

```
-----BEGIN PGP MESSAGE-----
Version: Mailvelope v2.1.1
Comment: https://www.mailvelope.com

9Spc/ZueJ+/+iO2kYfq0zi2whoX05RA4RM9cka8VyYGzTEOYpdRfcevFCGbO
KjqpbkNViBSlpY/S3m4GCUpfv1HR/WdSwx28IOolkeD7nKfXE0S/0OhmIaj3
```

Baoqiang Yan <byan@missouriwestern.edu>
350FF8164AEF16E9

Please enter your key password to decrypt this message.

Figure 2: Mailvelope Email Encryption and Decryption using RSA

The instructor uses small numbers to manually explain how to generate keys for RSA and how to encrypt/decrypt in class. Cryptool [2], instead, is employed for demo with large keys. Mailvelope [4], a plugin to Chrome that provides cryptographic service for many webmail clients, is a handy tool for students to gain real world experience of enhancing their email security with RSA. User can have their public key distributed via a public centralized key server and keep the private key local. As shown in figure 2, when composing a new message, the recipient's email turns green if found registered on the same server. When the recipient receives the encrypted email, the local private key is password protected to add another layer of security.

2.3 Matrix Math for Computer Graphics Transformation

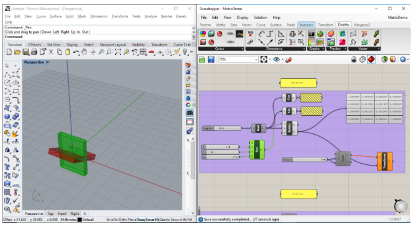


Figure 3: Rotation matrix demo in Rhino 3D and Grasshopper

Matrices are widely used in computer graphics calculations one of which is transformations. We focus on the basic reversible linear transformations such as scaling, shearing, translation, rotation in both 2D and 3D spaces, all done through multiplication with some homogeneous transformation matrix. Complex transformations can be realized through a series of basic ones.

Before dabbling in matrix operations, students need to understand vectors and their operations such as addition, subtraction, dot and cross products, especially their geometric meanings. The instructor uses Rhino 3D and Grasshopper for that purpose and to demonstrate the visual effects of different transformations such as rotation as shown in figure 3. Each transformation's parameter is configurable meaning the user can change the values in the transformation matrices by dragging bars. For example, in figure 3, user can change the rotation angle and the transformation matrix is updated correspondingly. The same transformation matrix can be applied to multiple visual objects.

After students get familiar with the relation between matrix math and graphics transformations, they are given an opportunity to put matrix operations in code using Easel [7]. Easel is a simple 3D graphics pipeline implementation written in Java targeted toward undergraduate Computer Graphics education. The instructor takes out some code about matrix operation to create an enticing assignment that, if students fill in the missing parts successfully, will show animated triangulated objects as shown in figure 4.

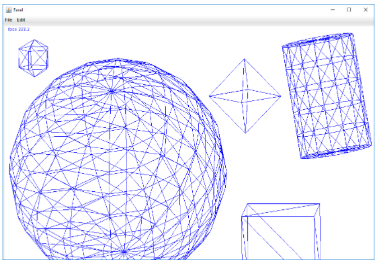


Figure 4: Animated 3D Objects in Easel

2.4 Probability Distribution Models for Queuing Simulation

Discrete event simulation of queuing system can help with resource optimization in many fields. There may be various random events involved in the simulation that we need to pick a Probability Distribution Model (PDM) for so that random values can be drawn for these events. There are two ways of determining which PDM to use, either by researching the features of different distribution models to see which types of events they are suitable for or by observing the histogram of actual collected event data to see which PDM's theoretical density function it resembles most. After the distribution model is selected for a random event, the required parameters need to be specified. They are normally based upon the tally results of the real collected data for the simulated event.

R studio comes in handy to help students better understand different PDMs and their relations. We start with Uniform Distribution and generalize to Bernoulli Distribution, Geometric Distribution, Binomial Distribution, Poisson Distribution, Exponential Distribution, Gamma Distribution. For each distribution, the instructor demonstrates the corresponding R functions to draw random values and their expected parameters. The Borel's law of large numbers is also verified for each PDM by comparing the actual simulated random drawing's histogram with theoretical probabilities.

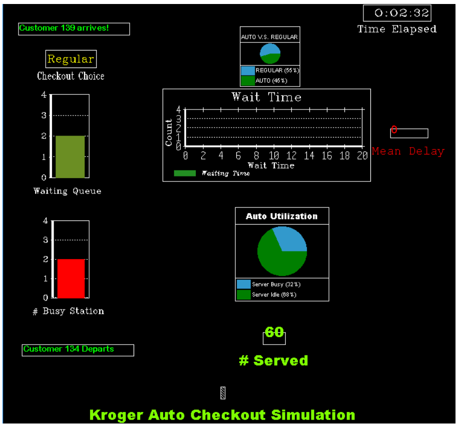
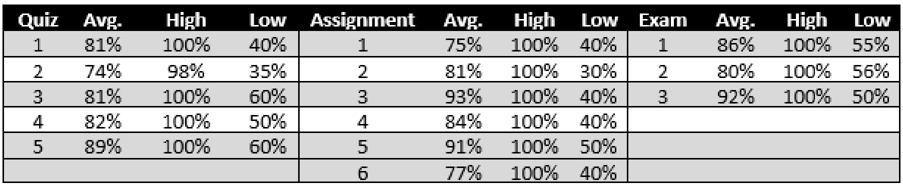


Figure 5: Simulation of Checkout at Kroger with SimStudio III

SimStudio with SimScript III [1] is used to show students how PDMs can be used in the simulation of a real-world queuing system – checking out at Kroger, a popular grocery store in southern states. Though having a relatively high learning curve, SimStudio III can visualize the whole simulation process as shown in figure 5. This makes study of simulation more enticing.

The goal of Kroger checkout simulation is to figure out the ideal number of auto checkout stations besides a regular cashier. The ideal number would bring the maximum utilization rate – percentage of time when these auto stations are in use. Every time the simulation starts, user can specify how many auto checkout stations to test. Based upon the tally results of the actual data observed at Kroger, customer interarrival time is determined to follow exponential distribution and the service time follows Gamma distribution with a shape parameter k that can be specified accordingly. Each customer is equally likely to use the cashier or an auto checkout station. This simulation can be run multiple times with different numbers of auto stations. The one that has the highest utilization rate and does not prolong the average waiting time would be the best to choose.

3 Assessment Results



Quiz	Avg.	High	Low	Assignment	Avg.	High	Low	Exam	Avg.	High	Low
1	81%	100%	40%	1	75%	100%	40%	1	86%	100%	55%
2	74%	98%	35%	2	81%	100%	30%	2	80%	100%	56%
3	81%	100%	60%	3	93%	100%	40%	3	92%	100%	50%
4	82%	100%	50%	4	84%	100%	40%				
5	89%	100%	60%	5	91%	100%	50%				
				6	77%	100%	40%				

Table 1 Overall Assessment Results via Quizzes, Assignments and Exams

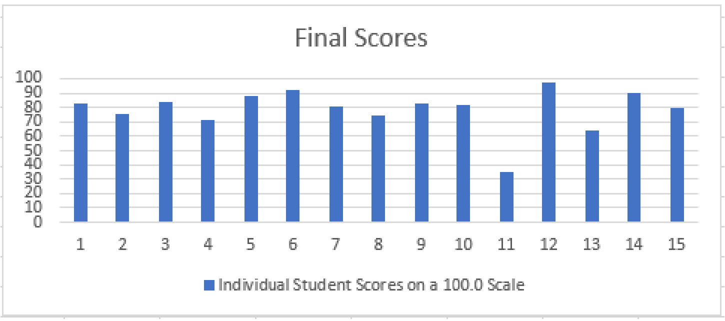


Figure 6: Final Score Distribution

As mentioned earlier, students are assessed via quizzes, assignments and exams. Quizzes are typically given about one week after a new topic is introduced to check student's understanding of the basic math concepts. Each quiz is given online and composed of ten multiple choice questions, short answer questions and essay questions. Assignments are given after the instructor talks about how the math concepts can be used in the corresponding CS applications mentioned in section 2. These assignments allow students to show their programming skills by implementing applications with the learned concepts based upon the samples demonstrated in class. Students can gain a sense of accomplishment for having successfully written a computer program. Exams are given when a topic is completely covered. The problems in the exams are comprehensive and closely related to the practical applications as well.

Table 1 shows the averages, high and low scores in all quizzes, assignments and exams. Though there are one or two students out of totally 15 students failing in some quizzes, assignments or exams, there are also some students getting 100%. The averages in table 1 and the distribution of individual final scores in figure 6 imply that this course was overall successful in conveying key math concepts while integrating them with applications.

4 Conclusion

This paper discusses the author's experience of teaching math for computer science at an open-enrollment university that has a mission statement of applied learning. To smooth the transition in student's mind from abstract math concepts to their inspiring applications, our CS program offers a special math course with four topics that are tied with well-crafted applications in computer science. The author discusses why the selected applications are useful in real world and how the math concepts in each topic are used in those applications. Finally, the assessment results of student's performance reveal that this course was overall successful.

References

[1] CACI. *SIMSCRIPT III Programming Manual.* CACI Products Company, 2007.

[2] Ctyptool. Ctyptool 2. `https://www.cryptool.org/en/cryptool2`.

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

[4] Mailvelope GmbH. Mailvelope. `https://www.mailvelope.com/en/`.

[5] Kenneth C. Louden. *Compiler Construction: Principles and Practice.* PWS Publishing Co., Boston, MA, USA, 1997.

[6] Christof Paar and January Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners.* Springer, 2010.

[7] Philip J. Rhodes and Baoqiang Yan. Easel: A Java based top-down approach to 3D graphics education. *Eurographics*, 2009.

Utilizing Deep Neural Networks for Brain–Computer Interface-Based Prosthesis Control*

Thomas C. Noel and Brian R. Snider
Department of Electrical Engineering & Computer Science
George Fox University
Newberg, OR 97132
{tnoel15,bsnider}@georgefox.edu

Abstract

Limb amputations affect a significant portion of the world's population every year. The necessity for these operations can be associated with related health conditions or a traumatic event. Currently, prosthetic devices intended to alleviate the burden of amputation lack many of the premier features possessed by their biological counterparts. The foremost of these features are agility and tactile function. In an effort to address the former, researchers here investigate the fundamental connection between agile finger movement and brain signaling. In this study each subject was asked to move his or her right index finger in sync with a time-aligned finger movement demonstration while each movement was labeled and the subject's brain waves were recorded via a single-channel electroencephalograph. This data was subsequently used to train and test a deep neural network in an effort to classify each subject's intention to rest and intention to extend his or her right index finger. On average, the employed model yielded an accuracy of 63.3%, where the most predictable subject's movements were classified with an accuracy of 70.5%.

*Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Every year approximately 185,000 limb amputations occur in the United States of America [11]. The loss of a limb can be life changing; once menial activities quickly become laborious and nontrivial. Limb loss can also be accompanied by mental health issues. Those who have recently lost a limb often report depression associated with the new impairment of mobility [8]. The loss of a limb can drastically change one's lifestyle and can be extremely discouraging.

The difficulty with adjustment to life after limb loss is directly related to loss of capability, both agile and tactile. What an amputee's lost arm was once able to feel and achieve through the intricacies of the biological brain-arm network is lost. The once unacknowledged, seamless tie between the arm and the brain becomes something mourned, especially in light of the sensation of the limb's persisting presence. Many amputees report feeling the limb's continued presence even after the limb is gone [8]. This sensation is an artifact of the brain's neuronal mapping associated with limb movement. These sensations can be painful or painless and can include sensations such as perception of movement, touch, temperature, pressure, vibration, and itch [15]. So the brain still possesses a capacity to process these stimuli that were once associated with the limb even after it is gone. This phenomenon seems to suggest that brain still possesses the toolkit necessary to re-establish connection with a limb, organic or not, in the place of the lost limb, given that the limb possesses compatible sensing and actuating mechanisms.

Efforts to develop such a device could be established by first developing rudimentary technologies that accurately interpret user intent. An attempt is here made to procure and investigate the viability of such a technology capable of detecting user intent to rest and extend his or her right index finger through the use of single-channel electroencephalograph (EEG) and intelligent classification of intention via a deep neural network (DNN).

2 Background

2.1 Brain–Computer Interfaces

There currently does not exist a prosthesis that can accurately and precisely report sensations and execute actions to the same functional degree as an organic human appendage. This is largely because of the limitations of today's brain-computer interfaces (BCI). The human brain possesses approximately 100 billion neurons, a number that makes the prospect of reading from each individual neural pathway in parallel daunting [6]. There has been a notable trend in BCI development over the past few decades, specifically, every seven years, the number of neural pathways that can be simultaneously monitored using a brain-computer interface doubles. Even with this exponential growth, progress is still currently slow; according to this model, reaching a point where all 100 billion neural pathways can be read simultaneously will take electrophysiologists nearly 220 years [16]. While this does provide hope for the amputees of tomorrow, it does beg the question of whether or not an alternative is currently in reach for amputees today. If such an alternative to total neural monitoring does exist it would surmount one of the biggest obstacles on the path to a more "seamless" prosthesis that senses and actuates with high levels of accuracy and precision.

Approaches toward the development of advanced prostheses and a greater understanding of the interaction between brain and limb have been rife. One such study sought to investigate the viability of functional electrical stimulation, through multi-electrode arrays implanted in the motor cortices of two rhesus macaque monkeys who underwent temporary limb paralysis. This method worked with the monkeys being able to control the flexion of four of their forearm muscles. During these trials, the monkeys effectively doubled their maximum voluntary wrist flexion force and were able to follow visually displayed force targets at two-thirds the speed of an unimpaired subject [13]. Studies such as this one support the viability of invasive BCIs (i. e., electrocorticography, or ECoG) for use in the prosthetics of tomorrow. Further studies utilizing invasive BCIs have exhibited that the motor cortex can form a stable neural representation for neuroprosthetic control, meaning that the brain exhibits a deft capacity for adapting to and cooperating with prostheses through the use of ECoG [5]. In one notable study, three test subjects (one with a neuroprosthetic arm) were trained to control the amplitude of beta rhythm recorded over the frontal areas of the brain using EEG. After six months of regular training, subjects were able to use these controlled signals to move a cursor to targets on a computer screen with greater than 90% accuracy. Additionally, the subject possessing a neuroprosthesis was able to use these signals to effectively grasp objects with his prosthetic arm [7]. In contrast to the six month training period that subjects underwent in the above study, another study utilizing EEG included the recording and power spectral analysis of neural signals from a single subject with an implanted neuroprosthesis over a three day training period. During this short time the subject was able to develop a stable neural representation that allowed him to consciously switch between grasp phases of the lateral grasp that his prosthetic provided. Using this developed ability, he was ultimately able to move a simple object from one place to another [10]. In other studies, researchers have used EEG signal mapping to send appropriate RF command signals to a prosthetic hand or have utilized support vector machines (SVM) to accurately predict the right or left-handedness of intended hand movement in subjects [12, 1].

2.2 Deep Neural Networks

The first computational model for an artificial neural network was presented by Warren McCulloch and Walter Pitts in 1943 [9]. In 1958, Frank Rosenblatt built on top of this work to develop the perceptron, an algorithm for pattern recognition [14]. Paul Werbos later developed a backpropagation algorithm that allowed these perceptrons to be layered, ultimately yielding the rudimentary model used for computing with artificial neural networks today [17]. A more modern major development in this field has been the development of deep learning, an idea first introduced in 1986 by Rina Dechter [4].

Recently there has been a resurgence of deep learning because of its uncanny in ability to classify data such as images and speech compared to more classical classification methods such as the SVM [3]. In an effort to capitalize on this machinery's ability to interpret digital signal data, methods are here employed in an effort to isolate and detect subjective intent based on brainwave signals collected from the subject's scalp.

3 Methodology

3.1 Data Collection

This study utilized electroencephalograms from five right-handed subjects, four male, one female. Each subject was connected to an EEG device, the Biopac MP36, via a single channel and had his or her brainwaves subsequently recorded for five trials, each three minutes in length. The electrodes were affixed to each subject's scalp using Elefix conductive EEG paste at F_Z, C_3, and C_4, as seen in Figure 1. This configuration choice was based on existing literature regarding optimal EEG electrode placement for the detection of subjective hand movement [2].

During each trial, subjects were told to mimic a video displaying a moving right index finger. The index finger executed one event per second in a pre-defined, looping sequence of events. As the subject replicated the movements of the right index finger on-screen, each event was automatically labeled in time. The sequence of finger-movement events used during this experiment was rest (R), rest to extension (RE), extension (E), extension to rest (ER), rest (R), rest to flexion (RF), flexion (F), and flexion to rest (FR). Because of the apparent doubled concentration of R events, only half of these, Rs preceding REs, were retained for final experimental analysis. This slight modification ensured that all events were equally represented in the training set, discussed in the next section. All data was collected in accordance with the collection procedure approved by our university's institutional review board for human subjects research.

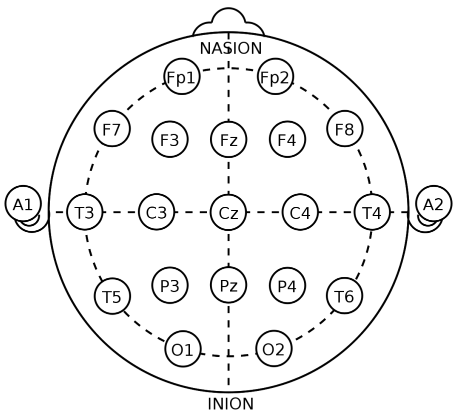


Figure 1: Standard EEG electrode placement; electrodes were placed at F_Z, C_3, and C_4.

3.2 Data Analysis

Signal data was collected at 500 Hz and events were labeled in the Biopac proprietary data analysis software, then subsequently exported and fed into a Python script where the signals were graphed and analyzed. Subsequently, the power spectral density (PSD) of each was calculated and plotted for exploration. In order to eliminate unnecessary information in the PSD data, a random forest classifier (RFC) was employed to rank PSD data points in order of significance. Using this information, researchers found that the frequency band that lent the most insight into subject intention was from approximately 12.76 Hz to 30.85 Hz. This information was used to inform which elements in the PSD data vector would be retained for training and testing the learning models.

Figure 2 depicts the retained portion of the power spectral density of both events. This retained portion was then processed using principal component analysis (PCA) to further reduce the data down to two dimensions, a feature vector size that was found to yield the best prediction performance. These labeled feature vectors were then used to train and test an SVM employing a radial basis function, and then a DNN utilizing the topology depicted in Figure 3. Training and testing was executed using k-fold cross validation where, for each subject, each model was trained on four of the subjects trials and then used to predict the held-out trial. The average accuracies of each of these classifications can be seen in Table 1 of the results section.

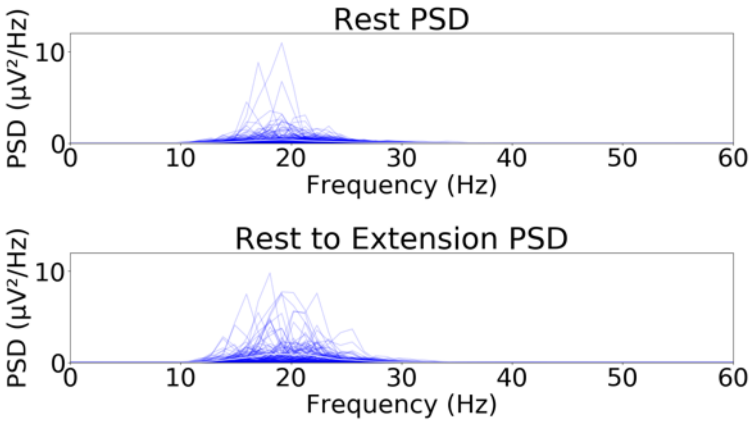


Figure 2: Filtered power spectral density of the subject brain signals.

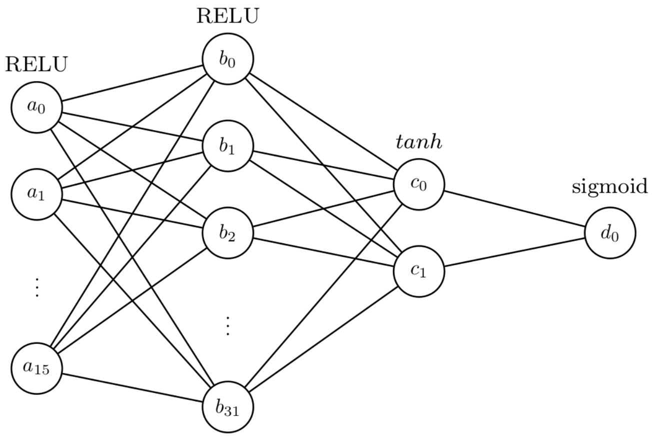


Figure 3: DNN topology with RELU input, RELU and *tanh* densing, and sigmoid output layers.

4 Results

Table 1 depicts the prediction accuracies generated by the DNN and the SVM. The average DNN classification accuracy across subjects was 63.3% and the average SVM classification accuracy across subjects was 62.4%. The best prediction accuracy across subjects was for the classification of subject B's inten-

Subject	Classifier	
	DNN	SVM
Subject A	0.600	0.586
Subject B	0.705	0.714
Subject C	0.627	0.609
Subject D	0.570	0.570
Subject E	0.664	0.641
Mean	0.633	0.624

Table 1: Per-subject and mean classification accuracy by classifier type.

tions where the DNN achieved an accuracy of 70.5% and the SVM achieved an accuracy of 71.4%. The t-value associated with these results was $t = 1.52$ and the p-value was $p = 0.203$, so the predictive accuracies of the SVM and DNN were not statistically significantly different from each other.

5 Conclusions and Future Work

Due to the complexity of this problem and the minimalist approach to brain signal sensing employed, the less-than-ideal results of this project were not entirely surprising. While the method of detecting subjective intention to rest or extend one's finger used here may not be practical, the results of this experiment beckon several other approaches to be explored in future work. These include incorporating the use of electrocardiography and oculography channels for artefact removal, adding more EEG channels, or utilizing an action, such as wrist flexion, that evokes a greater activation potential and repeating the process described here once more.

The implications of EEG-based intention detection beyond basic prosthetics are far-reaching. If further work reveals that non-invasive EEG monitoring can reliably yield subject intention or specific brain activity, technologies could be developed that support BCI-based control of mechanical and electrical systems. This would enable smart home network technology that would allow quadriplegic individuals to be able to perform household tasks such as opening doors, using the restroom, cooking, and cleaning without the need of human assistance.

Further research into BCI-based detection of other parameters describing an individual's state could be utilized to promote human safety. For example, driver wakefulness could be monitored to prevent traffic accidents by providing drowsiness warnings. Additionally, such a technology could be used by physicians to telemetrically monitor patient health. As medicine continues to become a more data-oriented profession, such a monitoring system could prove to be an invaluable diagnostic tool. If this technology were to be effectively harnessed, it would have the potential to revolutionize assistive and medical technology and drastically impact the way that humans and machines typically interact.

Acknowledgements

The researchers would like to thank the Paul K. Richter and the Evalyn E. C. Richter Memorial Funds for funding this research project and providing the opportunity to dig into this fascinating, emerging new field of research at the intersection of artificial intelligence, digital signal processing, and neuroscience.

References

[1] Bright, D. and Nair, A. and Salvekar, D. and Bhisikar, S. EEG-based brain controlled prosthetic arm. In *Proceedings of the 2016 Conference on Advances in Signal Processing (CASP)*, pages 479–483, 2016.

[2] Choi, S. H. and Lee, M. and Want, Y. and Hong, B. Estimation of Optimal Location of EEG Reference Electrode for Motor Imagery Based BCI Using fMRI. In *Proceedings of the 2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006.

[3] Ciresan, D. and Meier, U. and Schmidhuber, J. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[4] Dechter, R. Learning while searching in constraint-satisfaction-problems. 1986.

[5] Ganguly, K. and Carmena, J. M. Emergence of a Stable Cortical Map for Neuroprosthetic Control. *PLOS Biology*, 7(7), 2009.

[6] Herculano-Houzel, S. The Human Brain in Numbers: A Linearly Scaled-up Primate Brain. *Frontiers in Human Neuroscience*, 3(31), 2009.

[7] Lauer, R. T. and Peckham, P. H. and Kilgore, K. L. EEG-based control of a hand grasp neuroprosthesis. *NeuroReport*, 10:1767–1771, 1999.

[8] Maguire, P. and Parkes, C. M. Surgery and loss of body parts. *The BMJ*, 316:1086–1088, 1998.

[9] McCulloch, W. and Pitts, W. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:111–113, 1943.

[10] Muller-Putz, G. R. and Scherer, R. and Pfurtscheller, G. and Rupp, R. EEG-based neuroprosthesis control: a step towards clinical practice. *Neuroscience Letters*, 382:169–174, 2005.

[11] Owings, M. and Kozak, L. J. Ambulatory and Inpatient Procedures in the United States. *Vital and Health Statistics*, 13(139):8, 1998.

[12] Pfurtscheller, G. and Flotzinger, D. and Mohl, W. and Peltoranta, M. Prediction of the side of hand movements from single-trial multi-channel EEG data using neural networks. *Electroencephalography and Clinical Neurophysiology*, 82:313–315, 1992.

[13] Pohlmeyer, E. A. and Oby, E. R. and Perreault, E. J. and Solla, S. A. and Kilgore, K. L and Kirsch, R. F. and Miller, L. E. Toward the Restoration of Hand Use to a Paralyzed Monkey: Brain-Controlled Functional Electrical Stimulation of Forearm Muscles. *PLOS One*, 4(6), 2009.

[14] Rosenblatt, F. The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain. *Psychological Review*, 65:386–408, 1958.

[15] Rugnetta, M. *Phantom limb syndrome.* Encyclopædia Brittanica, Inc., 2014.

[16] Stevenson, I. H. and Kording, K. P. How advances in neural recording affect data analysis. *Nature Neuroscience*, 14:139–142, 2011.

[17] Werbos, P. Beyond regression: new tools for prediction and analysis in the behavioral sciences. 1974.

Introduction to Jetstream: A Research and Education Cloud[*]

Conference Tutorial

Sanjana Sudarshan and Jeremy Fischer
Research Technologies
Indiana University
Bloomington, IN 47401
{ssudarsh, jeremy}@iu.edu

1 Introduction

Jetstream is the first production cloud funded by the National Science Foundation (NSF) for conducting general-purpose science and engineering research as well as an easy-to-use platform for education activities. Unlike many high-performance computing systems, Jetstream uses the interactive Atmosphere graphical user interface developed as part of the iPlant (now CyVerse) project and focuses on interactive use on uni-processors or multiprocessors. This interface provides for a lower barrier of entry for use by educators, students, practicing scientists, and engineers. A key part of Jetstream's mission is to extend the reach of the NSF's eXtreme Digital (XD) program to a community of users who have not previously utilized NSF XD program resources, including those communities and institutions that traditionally lack significant cyberinfrastructure resources. One manner in which Jetstream eases this access is via virtual desktops facilitating use in education and research at small colleges and universities, including Historically Black Colleges and Universities (HB-CUs), Minority Serving Institutions (MSIs), Tribal colleges, and higher education institutions in states designated by the NSF as eligible for funding via the Established Program to Stimulate Competitive Research (EPSCoR).

While cloud resources won't replace traditional HPC environments for large research projects, there are many smaller research and education projects that would benefit from the highly customizable, highly configurable, programmable

[*]Copyright is held by the author/owner.

cyberinfrastructure afforded by cloud computing environments such as Jetstream. Jetstream is a Infrastructure-as-a-Service platform comprised of two geographically isolated clusters, each supporting hundreds of virtual machines and data volumes. The two cloud systems are integrated via a user-friendly web application that provides a user interface for common cloud computing operations, authentication to XSEDE via Globus, and an expressive set of web service APIs.

Jetstream enables on-demand access to interactive, user-configurable computing and analysis capability. It also seeks to democratize access to cloud capabilities and promote shareable, reproducible research. This event will describe Jetstream in greater detail, as well as how its unique combination of hardware, software, and user engagement support the "long tail of science." This tutorial will describe Jetstream in greater detail, as well as how its unique combination of hardware, software, and user engagement support the "long tail of science." Attendees will get a greater understanding of how Jetstream may enhance their education or research efforts via a hands-on approach to using Jetstream via the Atmosphere interface.

2 Tutorial Description

This tutorial requires two to three hours.

- Prerequisites: Basic Linux command line knowledge a plus (but not required)
- Required: Laptop, modern web browser (Chrome, Firefox, Safari)
- Targeting: Educators, Researchers, Campus Champions/ACI-Ref Facilitators, Campus research computing support staff

This tutorial will first give an overview of Jetstream and various aspects of the system. Then we will take attendees through the basics of using Jetstream via the Atmosphere web interface. This will include a guided walk-through of the interface itself, the features provided, the image catalog, launching and using virtual machines on Jetstream, using volume-based storage, and best practices.

We are targeting users of every experience level. Atmosphere is well-suited to both HPC novices and advanced users. This tutorial is generally aimed at those unfamiliar with cloud computing and generally doing computation on laptops or departmental server resources. While we will not cover advanced topics in this particular tutorial, we will touch on the available advanced capabilities during the initial overview.

3 Tutorial Program

This is a sample tutorial program. Time required for this tutorial is approximately 3 hours.

- What is Jetstream?
- Q & A and what brief hands-on overview
- Getting started with Jetstream, including VM launching
- Break
- Accessing your VM, creating and using volumes
- Customizing and saving images, DOIs
- Cleaning up
- Final Q & A

Teaching Introduction to Programming Courses to Non-Computer Science Majors using SageMath*

Conference Tutorial

Razvan A. Mezei
Computer Science Department
Saint Martin's University, Lacey, WA 98503
`rmezei@stmartin.edu`

In this tutorial we will demonstrate the use of SageMath in Introduction to Programming courses for non-Computer Science majors. SageMath (which is Python-based) is a great tool both for programming ([3]) and symbolic computation ([1, 4]). Being a free open source alternative to Matlab, Mathematica and Maple ([5]), it could be used to teach introductory programming courses ([3]), especially to students with an interest in Mathematics, Teaching Education, and Data Science. The tutorial will cover topics such as: various ways to access SageMath (host-based, web-based, or cloud-based), an overview of symbolic computation in SageMath, typical topics seen in Introductory Programming courses ([2, 6]), as well as how to program Interacts and Animations in SageMath. We will end this tutorial with a discussion on how to help students contribute to this open source project.

References

[1] G.V. Bard. Sage for undergraduates (online version). `http://www.gregorybard.com/sage.html`.

[2] Tony Gaddis. *Starting Out with Python (4th Edition)*. Pearson, 2017.

[3] R. A. Mezei. Teaching an introductory programming course to non-computer science majors using sagemath. *Turk. J. Math. Comput. Sci.* 11(1)((2019)) 24–28.

[4] W.A. Stein et al. Collaborative calculation in the cloud. `https://cocalc.com`.

[5] W.A. Stein et al. SageMath - open-source mathematical software system. `http://www.sagemath.org/`.

[6] John Zelle. *Python Programming: An Introduction to Computer Science, 3rd Ed.* Franklin, Beedle & Associates, 2016.

*Copyright is held by the author/owner.

Applying Code Translation and Subprogram Call Graph to Improve Programming Proficiency in CS1[*]

Conference Tutorial

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

Abstract

This tutorial introduced the application of two techniques: code translation and subprogram call graphs, helping the students, especially those in CS1, to improve their programming proficiency. It started by a brief introduction of each technique, followed by the examples of their application in class, and finally the sample exercises, assignments, and quizzes are presented.

1 Introduction

When students study a programming language, especially for CS1 students, they can face many challenges such as:

- When reading the code, a student can encounter unfamiliar syntax, because of which the student will not be able to correctly understand the meaning of the code.
- When reading the code of a complicated program, students often can be confused by the meaning of the code. Therefore, they often need to interpret the meaning of the code into human nature languages in mind, and then understand the code according to the meaning in human nature languages.

[*]Copyright is held by the author/owner.

- When writing the code, a student will experience the situation that what they want to operate can be clearly expressed in a human nature language, but they cannot find the identical expressions in a programming language. As the result, the code to be written cannot be clearly and concisely expressed the operations that the program is assumed to do.
- When compiling the code, though students can understand that the program has made certain mistakes based on the error messages from the compiler, they cannot quickly determine the reasons so as to correctly debug.

There are many reasons for these challenges, one of which is because of unfamiliar with the syntax of a programming language. Hence, assisting the students thoroughly learn the syntax of a programming language and thereby deeply understand the code become vital.

When learning a foreign language for example English, students often encounter challenges similar to those in learning a programming language. There are many similarities between programming languages and human nature languages, so according to the author's experience, the method of learning a foreign language, like the translation exercises, can also be applied in the programming exercise, helping the students to better understand the syntax. That is, when learning a programming language, the exercises like writing down the meaning of each line of code in the form of comments, or writing the code based on given comments can be extensively applied in different way.

The translation exercise are effective for the students to be familiar with the syntax of a programming language, but it often does not work well for the students to understand the calling sequence among the functions in a program. A common way to comprehend an article when learning a foreign language is to, before deeply understand the meaning of each paragraph, identify the structure of an article and the general meaning of each paragraph. Such a method can make the students better understanding the interrelationship among paragraphs and the meaning of the entire article. This technique, when learning the programming language, can be implemented with a subprogram call graph similar to a dynamic flow graph. That is, before deeply understand the code of each function, a subprogram call graph is used to analyze and represent the call relationships among functions, parameter passing, and return values, thus helping the students find out the relationship between functions of a program. Then, the translation exercise can be selected to focus on the meaning of each line of the code in each function. In practice, these two techniques can be used separately or together and designed as various kinds of exercises, some of which will be introduced in this tutorial.

A Comparison of Two Hands-On Cybersecurity Frameworks[*]

Conference Tutorial

Jens Mache[1] *and Richard Weiss*[2]
[1]*Lewis & Clark College*
Portland, OR 97219
{jmache,author}@lclark.edu
[2]*The Evergreen State College*
Olympia, WA 98505
{weissr,author}@evergreen.edu

Abstract

There are several different frameworks for teaching hands-on cybersecurity exercises. Faculty who want to integrate cybersecurity into their courses may have difficulty in choosing one. In this tutorial, we will show faculty two different frameworks so that they can understand the possibilities. It is not necessary to choose only one. In our courses, we have taken advantage of multiple frameworks, in order to benefit from their individual strengths. We think this will lower the barrier for use.

In this tutorial, we will introduce the DeterLab and EDURange frameworks, and present one hands-on exercise from each. Participants try them, and discuss how they can be used in their courses.

1 Overview

Student exposure to practical, hands-on exercises is critical for cybersecurity curricula. It helps students internalize concepts taught in class, learn to use cybersecurity tools, and learn critical and adversarial thinking.

EDURange [1, 4, 5, 6, 7, 8] is a cloud-based framework for cybersecurity exercises designed with three major goals. First, ease-of-use for students and instructors. Scenarios run on VMs that are created automatically in the public cloud. Students don't need special software and can work anywhere with

[*]Copyright is held by the author/owner.

Internet service. Instructors can register their classes. Students can work in groups. EDURange collects data to make assessment easier. Second, engaging for students and faculty. Students from a variety of backgrounds can learn practical security concepts, tools, and skills in scenarios that gamify realistic challenges. Third, flexibility. Use simple scripts to specify exercises at a high level and create variations. This enables instructors to tailor exercises to their specific classes and student backgrounds and continue to modify them in order to minimize risk of students finding the answers online.

DeterLab [2, 3] is both an educational and a research platform on a private cloud. Once the instructor reserves resources for their class, students have control over starting and stopping their "experiment". Similar to EDURange, there are scripts that install the OS and required software packages. DeterLab has a variety of "homework" exercises, available via the education portal [2], cover a wide range of topics including: buffer overflows, code injection and command-injection attacks, man-in-the-middle attacks, worm modeling and detection, botnets, router and DNS attacks, and DDoS attacks. Each of these exercises is a packaged experiment that demonstrates one of these topics, providing students with direct observation of attacks and interaction with targets. Students create and manipulate an instance of an experiment and follow the instructions to demonstrate attacks and defenses, and improve their practical cybersecurity skills. Both DeterLab and EDURange rely on the command line interface and have the ability to capture and analyze student interactions.

2 Acknowledgements

This work was partially supported by National Science Foundation grants 1723705, 1723714, 1516100 and 1516730.

References

[1] `https://edurange.org/scenarios.html`, accessed July 2019.

[2] `https://www.isi.deterlab.net/sharedpublic.php`, accessed July 2019.

[3] J. Mirkovic and T. Benzel. Teaching Cybersecurity with DeterLab. *IEEE Security Privacy*, 10(03):73–76, 2012. doi: `https://doi.ieeecomputersociety.org/10.1109/MSP.2012.23`.

[4] R. Weiss, S. Boesen, J. Sullivan, M. E. Locasto, J. Mache, and E. Nilsen. Teaching cybersecurity analysis skills in the cloud. In *Proceedings of the 46th ACM Technical Symposium on Computing Science Education*,

SIGCSE '15. ACM, 2015. doi: `http://dx.doi.org/10.1145/2676723.2677290`.

[5] R. Weiss, M. E. Locasto, and J. Mache. A reflective approach to assessing student performance in cybersecurity exercises. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 597–602. ACM, 2016. doi: `http://dx.doi.org/10.1145/2839509.2844646`.

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

[7] R. Weiss, F. Turbak, J. Mache, and M. E. Locasto. Cybersecurity education and assessment in EDURange. *IEEE Security Privacy*, 15(3):90–95, 2017. doi: `http://doi.ieeecomputersociety.org/10.1109/MSP.2017.54`.

[8] R. Weiss, F. Turbak, J. Mache, E. Nilsen, and M. E. Locasto. Finding the balance between guidance and independence in cybersecurity exercises. In *USENIX Workshop on Advances in Security Education*, 2016.