The Journal of Computing Sciences in Colleges

Papers of the 23nd Annual CCSC Northwestern Conference

October 8-9, 2021
Saint Martin's University
Lacey, WA

Baochuan Lu, Editor
Southwest Baptist University

Sharon Tuttle, Regional Editor
Humboldt State University

Volume 37, Number 1 **October 2021**

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

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

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

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

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

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

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

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

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

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

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

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

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

Shereen Khoja, Northwestern Representative(2021), shereen@pacificu.edu, Computer Science, 2043 College Way, Forest Grove, OR 97116.

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

Tina Johnson, South Central Representative (2021), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308.

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

Bryan Dixon, Southwestern Representative (2023), (530)898-4864,

bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

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

Bin Peng, Associate Editor, (816) 584-6884, bin.peng@park.edu, Park University - Department of Computer Science and Information Systems, 8700 NW River Park Drive, Parkville, MO 64152.

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

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

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

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

CCSC National Partners

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

Platinum Partner

Turingscraft
Google for Education
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Bronze Partners

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

Welcome to the 2021 CCSC Northwestern Conference

The 2021 Northwest Steering Committee is very pleased to welcome everyone to the Twenty First Annual CCSC Northwestern Conference hosted this year by Saint Martin's University.

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 accepted twelve papers, six tutorials, and student posters. The steering committee accepted twelve papers out of sixteen papers submitted (75%). All papers, panels, and tutorials went through the regular peer 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 Laurie White, Cloud Developer Relations Engineer at Google, begin our conference with her keynote address on "New Applications that Are now Possible on the Cloud".

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.

Mario Guimaraes
Saint Martin's University
Conference Chair

2021 CCSC Northwestern Conference Steering Committee

Mario Guimaraes, Conference Chair ..... Saint Martin's University
Xuguang Chen, Site Chair ..... Saint Martin's University
Bob Lewis, Program Chair ..... Washintgon State Universtity, Tri-Cities
Razvan Mezei, Papers Chair ..... Saint Martin's University
Gina Sprint, Panels Tutorials ..... Gonzaga University
Ben Tribelhorn, Partners Chair ..... University of Portland
Richard Weiss, Student Posters Chair ..... The Evergreen State College

Regional Board — 2021 CCSC Northwestern Region

Shereen Khoja, Regional Representative ..... Pacific University
Dan Ford, Treasurer ..... Linfield College
Sharon Tuttle, Editor ..... Humboldt State University
Shereen Khoja, Past Conf. Chair ..... Pacific University
Nadra Guizani, Next Conf. Chair ..... Washington State University
Clint Jeffery, Registrar ..... University of Idaho

Reviewers — 2021 CCSC Northwestern Conference

Ashish Aggarwal ..... Univ. of Florida, Gainesville, FL
Anthony Bowman ..... Univ. of Alabama at Birmingham, Birmingham, AL
Xuguang Chen ..... Saint Martin's Univ., Lacey, WA
Gabriel V. de la Cruz Jr. ..... North Idaho College, Coeur d'Alene, ID
Janet Davis ..... Whitman College, Walla Walla, WA
Zachary Dodds ..... Harvey Mudd College, Claremont, CA
Mario Guimaraes ..... Saint Martin's Univ., Lacey, WA
Amanpreet Kapoor ..... Univ. of Florida, Gainesville, FL
Robert R. Lewis ..... Washington State Univ., Richland, WA
Guangyan Li ..... Saint Martin's Univ., Lacey, WA
Shyam P. Prabhakar, Researcher and Ph.D. Student ... Univ. Of Alabama at Birmingham, Birmingham, AL
Richard Weiss ..... The Evergreen State College, Olympia, WA
Howard Whitston, Retired Instructor .. Univ. of South Alabama, Mobile, AL

Doing the (Almost) Impossible*

Keynote

Laurie White
CS Professor Emeritus, Mercer University
Developer Relations Engineer, Google Cloud
and former CCSC:SE Program Chair

The resources provided by computing are making tasks that once seemed impossible almost routine. And cloud computing makes these resources available to a far larger audience than ever before. This talk will present some of the (almost) impossible things that have been made possible and look at how they were accomplished. Ideally, you'll be dreaming of some (almost) impossible things you may be able to do by the end of the talk.

Expanding Career Pathways: The Joy of Teaching Computer Science at Predominately Undergraduate Liberal Arts Institutions*

Panel Discussion

Haiyan Cheng[1], Janet Davis[2], Shereen Khoja[3], Tammy VanDeGrift[4]
[1]Computer Science Department
Willamette University, Salem, OR 97301
hcheng@willamette.edu
[2]Computer Science Department
Whitman College, Walla Walla, WA 99362
davisj@whitman.edu
[3]Mathematics and Computer Science Department
Pacific University, Forest Grove, OR 97116
shereen@pacificu.edu
[4]Computer Science, Shiley School of Engineering
University of Portland, Portland, OR 97203
vandegri@up.edu

1 Panel Abstract

Four CS faculty members from different primarily undergraduate institutions (PUIs) will share the joys and challenges of such careers. The media has broadcasted the need for educating more students in computer science, especially PUIs with liberal arts [2, 3, 4]. In order to educate the growing CS student enrollment at PUIs [1], we need to expand and support the pipeline of potential faculty members to teach at PUIs. The primary audience for this panel includes graduate students pursuing a computing PhD and those with PhDs who are considering a career change. If the panel is virtual or offered remotely, it can be advertised to graduate students across the USA. The goals of this panel include: 1) provide an overview of joys and opportunities of PUI careers, 2) provide an overview of the challenges of this type of career, 3) invite other attendees to contribute from their experiences, and 4) answer attendees' questions, and 5) create a community of current and future PUI faculty among attendees and the panelists. In the sections that follow, we describe the joys and challenges of faculty careers at institutions that focus on undergraduate education.

1.1 The Joys

The PUI faculty member has three job responsibilities: teaching, research, and service. At a PUI, teaching is the primary responsibility with teaching loads of two to three classes or labs per semester. The panelists find joys in all three areas and, in many cases, the three areas intersect.

Working with undergraduates: First and foremost, the panelists enjoy working directly with undergraduate students in smaller settings. We enjoy teaching small classes, mentoring undergraduate research students, advising students to make the most of their goals in college and preparing them for their futures, observing the same student make transformations across several courses, and getting to know students as individuals. We get to engage with students in the classroom and outside the classroom. For example, the panelists advise student clubs, travel with students for study abroad, and attend conferences with undergraduate students.

Staying current: As faculty at smaller institutions, we get to teach a wide variety of classes that keeps us current about trends in computing and trends in computing education. In some cases, we get to teach first-year seminars and senior capstones that provide professional growth, collaboration, and more inter-disciplinary lenses. We often work within small departments (three to eight CS faculty members) and are the stewards of the CS curriculum, so updates to courses and the program can be made regularly and respond to the needs of the profession.

Personal growth and autonomy: Since the CS program is small, we have more autonomy in designing and delivering courses. We also engage with faculty outside of CS on a regular basis, through inter-disciplinary co-teaching and research opportunities. We enjoy getting to learn about many fields and think creatively about how CS can integrate with other disciplines. At smaller schools, there are fewer artificial walls between disciplines and there is strong community of faculty across campus. Research cycles are often short with undergraduates during a summer or semester. Faculty can use their summers for research, consulting, development, and restoration. Having the cadence of semesters means the cycle of work has firm deadlines.

1.2 The Challenges

As with any career, there are both opportunities and challenges. The panelists will offer some of the challenges of a PUI teaching career.

Small staff: Because departments are small, faculty must teach in areas outside of their training. While this can accelerate personal growth, too many new courses can be challenging. Recruiting, hiring, and retaining CS faculty at PUIs can be challenging due to small applicant pools. Administrative assistance is often shared across multiple departments. Senior faculty may have heavy service loads. When there is growth in student enrollments or when faculty have personal and sabbatical leaves, adjuncts or visiting professors must be hired.

Bending too much: Because PUIs are dedicated to serving undergraduates, faculty can get drawn into helping too much. Students may see faculty as on-call 24/7 for questions. Faculty can also get pulled into decisions around scheduling course times to work for all students, for example including student-athletes. We may bend too much, in some cases, to assist students - for example, offering independent studies or frequent meetings with advisees. Because we prioritize teaching, we must schedule personal obligations to work with the academic calendar.

Institution-type career choice: Movement between PUIs is possible; for example, one of the panelists has successfully moved from one PUI to another. Moving from a PUI to a research institution may be more challenging due to reduced scholarship production and grant awards at a PUI. PUIs have few faculty who must cover the entire curriculum, so it is unlikely to have a departmental colleague in the same research area, and it is often necessary to find research collaborators outside the institution.

Compared to research universities, PUIs provide a different path, and not a lesser path, to a faculty career.

References

[1] Computing Research Association. Generation cs: Computer science undergraduate enrollments surge since 2006, 2017. https://cra.org/data/Generation-CS/.

[2] B. Fung. Tech companies are hiring more liberal-arts majors than you think. https://www.washingtonpost.com/news/the-switch/wp/2015/08/26/tech-companies-are-hiring-more-liberal-arts-majors-than-you-think/.

[3] I. Kowarski. What can you do with a computer science degree? https://www.usnews.com/education/best-graduate-schools/articles/2019-05-02/what-can-you-do-with-a-computer-science-degree.

[4] A. Loten. America's got talent, just not enough in IT. https://www.wsj.com/articles/americas-got-talent-just-not-enough-in-it-11571168626.

Teaching Test-driven Development of Algorithms behind Data Science Library APIs*

Gina Sprint[1]
[1]Department of Computer Science
Gonzaga University
Spokane, WA 99258
sprint@gonzaga.edu

Abstract

Members of a growing community of data science and machine learning experts are advocating for the increased use of explainable and interpretable algorithms. To help students understand these algorithms, this paper presents a programming-intensive course, CPSC 322 Data Science Algorithms, that utilizes a modern data science technology stack to introduce students to fundamental and explainable data science algorithms. In this course, students utilize test-driven development (TDD) to implement data science algorithms "from scratch," while adhering to the same application programming interface (API) as the industry-standard library. With this approach to teaching data science, students learn popular library APIs by building their own mini version of the libraries. They also learn the "what" and the "why" behind the libraries instead of mostly using them as black boxes. Along the way, students in CPSC 322 also gain experience with software engineering tools in an industry-standard data science tech stack, including Docker, Git/Github, automated testing, and machine learning model deployment with Flask and Heroku. Over the duration of the course, students' experience with these tools, API programming, and TDD increase significantly.

1 Introduction

In our increasingly data-driven world, there is a growing need for data scientists with skills spanning several layers of a modern data science technology stack, including reproducibility infrastructure (e.g., Docker, virtual machines), command line, development support (e.g., Git, package managers), computational narrative and workflow (e.g., Jupyter, R Markdown), programming (e.g., Python, R), and data modeling and visualization libraries (e.g., pandas, tidyverse) [9]. Consequently, instructors in higher education and practitioner-instructors from industry are offering data science courses/bootcamps to train data scientists in these technologies. Since data science algorithms are becoming more complex, such as the implementation of deep neural networks, when teaching popular libraries, these curriculums typically focus on the libraries' application programming interface (API). This can be problematic because data scientists should also understand the algorithms they are using. Using algorithms properly depends on computer science (e.g., implementing, testing, debugging, deploying, and maintaining algorithms), mathematics (e.g., properly applying and interpreting statistical results), and business knowledge (e.g., accurately and effectively communicating insights and actions [4]).

To elaborate, the complexity of data science and machine learning algorithms often results in effectively treating the algorithms as black boxes (BBs). With a BB, the details of how its inputs are transformed into outputs are not known to the scientist, either due to its complexity or proprietary nature [11]. Since these complex models are increasingly being deployed for decision making in domains like healthcare and criminal justice, it is important that the models are explainable. Christopher Molnar, author of *Interpretable Machine Learning*, states that for some problems a correct output alone does not fully solve the problem, the algorithm should also explain how it came to the output [10]. This essentially complements the output's "what" with its "why." The interpretable machine learning community is growing and so is the public's desire for more transparency in data-driven decision making. In response, data science educators should aim to teach the details of fundamental algorithms to ensure students can explain the algorithms and outputs. To help address this need, this paper presents CPSC 322 Data Science Algorithms, a course that trains students in both the "what" and the "why" behind common libraries.

2 Related work

In 2019, Cynthia Rudin published a perspective article titled, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead" [11]. In this article, she argues that the machine

learning community should use "inherently interpretable" models instead of trying to explain BB models. For the latter, several techniques have been proposed and utilized to inspect BB machine learning models, such as interactive visual analytics systems [8]. For the former, educators and authors have explored how to teach the details of data science algorithms and libraries. For example, in *Data Science from Scratch: First Principles with Python*, author Joel Grus guides readers through implementing data science algorithms. He states the best way to learn data science is by "building tools and implementing algorithms by hand in order to better understand them" [5].

More specifically in higher education, educators in computer science, mathematics, and business have advocated for including more details and interpretability when teaching data science. Beginning with business education, Delibasic and colleagues conducted an experiment with 118 senior students in a business intelligence course [3]. The students were divided into two groups, one learning BB decision trees and the other using white box (WB) decision trees. Despite the increased complexity, WB students perceived the decision tree algorithms were easy to use, though it took them longer to improve accuracy. The authors recommend educators explain more algorithm structure and benefits by complementing a BB introduction with more WB tools. In mathematics education, Hardin and Horton advocated for the importance of discrete and continuous mathematics in data science education, stating "students who skip out on math completely run the peril of black box thinking" [6]. Since mathematics and statistics form the underpinnings of many data science algorithms and approaches, fast-tracking past these fundamentals means not understanding the uncertainties and limitations of an algorithm. Lastly, in computer science education, educators are calling for more formal software engineering in the training of data scientists. In 2017, Cruz described how code correctness, reliability, modifiability, testability, reusability, maintainability, and efficiency are important, yet lacking, in data science curriculums [2]. The author concludes that data science curriculums should at least include the areas of software architecture and design, process and management, and quality and testing. Similarly arguing the need for high quality software engineering in data science, author Matthew Kirk states that "writing successful machine learning code comes down to being disciplined enough to follow the principles of design. . . , and writing tests to support your code-based hypotheses" [7].

3 Methods

CPSC 322 Data Science Algorithms is an undergraduate, programming-intensive course with the primary goal of having students implement fundamental algorithms "from scratch" with minimal dependence on libraries. The pre-requisite

for the course is at least one semester of programming in Python or at least two semesters of programming in C++. The course was designed with a modern data science technology stack, including Docker, Git, pip, Jupyter, Python, and exposure to data science libraries. Topics of the course include an introduction to the field of data science, Git and Github for version control, Python basics (including object-oriented programming, packages, and unit testing), exploratory data analysis techniques (including cleaning/preparation, summary statistics, visualization, regression, and Latex/Markdown for Jupyter Notebooks), supervised machine learning, ensemble learning, unsupervised machine learning, APIs, and model deployment. The delivery of this content consists primarily of lab-style lectures where traces of the algorithms are performed live on an iPad, followed by a derivation of algorithm starter code.

Over the course of the semester, students are incrementally taught reproducibility infrastructure with Docker and test-driven development (TDD) using PyTest. For Docker, students are shown how to use a Docker container created from a specific Anaconda3 Python distribution image (e.g., the continuumio/anaconda3:2020.11 image). Using a Docker container environment offers students flexibility because they can use command line and/or IDE-based development (e.g., VS Code with the Remote Containers extension). Beyond infrastructure, reproducibility is also emphasized throughout the course, such as via discussions about seeding random number generators and as part of TDD. For TDD, students are first given unit tests written by the instructor and are shown how to run, debug, and pass the unit tests in class. Then, students learn how assert statements work and why they form the building blocks of unit tests. Next, students are given test cases and are challenged with writing their own unit tests. This in-class trajectory is paralleled in the homework assignments, called programming assignments (PAs). Table 1 provides an overview of algorithms implemented in the PAs. PAs involve a significant amount of programming, testing, and debugging of data science algorithms. For the first two PAs, instructor-written unit tests are provided for students. For the remaining PAs, students write their own unit tests based on test cases provided in class via iPad traces or in the textbook. The textbook for the course is *Principles of Data Mining*, which aims "to help general readers develop the necessary understanding of what is inside the 'black box' so they can use commercial data mining packages discriminatingly" [1].

Like Docker and TDD, additional software engineering methods are integrated throughout the course, including creating custom packages and implementing an automated testing workflow. For the workflow, students use Github Classroom to track changes in their code and to submit their PAs. On push to their PA repository, a Github Action is triggered that runs pushed code against unit tests for functional correctness. Additionally, the course includes how to

use APIs client-side to collect data and how to create the server-side of an API. To do this, students are shown how to design, implement, and deploy their own API endpoint for making machine learning predictions. This involves deploying a Flask app as a custom Docker container running on a Heroku dyno. Flask is a lightweight micro-web framework and Heroku is a platform-as-a-service that offers free web app hosting with the Github Student Developer Pack.

Table 1: Summary of the main algorithms implemented "from scratch."

Algorithm	**When implemented**?	**Unit tested?**
CSV file parsing	In-class; PA1	No
Common summary stats	In-class; PA1-2	Some
Handling missing values	In-class; PA1-2	Some
Common table operations (group-by, find duplicates, drop rows, shuffle, inner/outer join, etc.)	In-class; PA2	Some
Discretization and histogram creation	In-class; PA3	Yes
Simple linear regression	In-class; PA3	Yes
Hold out method	In-class; PA4	Yes
k fold and stratified k fold cross validation	PA4	Yes
Confusion matrix construction and evaluation	PA4	Yes
k nearest neighbors classification	PA4	Yes
ZeroR and random classification	PA5	Yes
Naive Bayes classification	PA5	Yes
TDIDT decision tree classification	PA6	Yes
Random forest w/bagging classification	Project	Yes
Apriori association rule mining	PA7	Yes
k means clustering	PA8 (bonus)	Yes

Over the course of the semester, the PAs require students to build a mini version of the Python sci-kit learn machine learning library via a Python package called `mysklearn`. The `mysklearn` package answers the "what" behind the libraries. It consists of several reusable classes, including a simple version of the pandas library's `DataFrame` class, several classifiers that follow sci-kit learn's `fit()` and `predict()` API for estimators, and several utility functions that implement common data science algorithms. Many of these utility functions follow the API of their industry-standard library counterpart, such as numpy, scipy, and pandas, and are unit tested against these libraries as well. An overview of the `mysklearn` algorithms implemented in class and as part of PAs is provided in Table 1. For example, PA6 has students implement decision tree classifier training using the top-down induction of decision tree

(TDIDT) algorithm and entropy. Given some starter code, students implement a unit test for TDIDT and the algorithm. Starter code for PA6 demonstrating `mysklearn`'s API is included in the Appendix. Once students pass the unit tests for a PA, they use their `mysklearn` package with real-world datasets, evaluating their algorithms and explaining their results in Jupyter Notebooks. The algorithm interpretation in the Notebooks helps answer the "why" behind the algorithm outputs. The course ends with a large project where pairs of students use TDD to implement a random forest classifier, evaluate classifiers on a dataset of their choosing, and deploy the "best" classifier to Heroku.

4 Results and Discussion

To evaluate the effectiveness of the course, students enrolled in the Spring 2021 course offerings (two sections, N = 30 in each section) were surveyed at the start (pre-course) and end of the semester (post-course). The pre-course survey asked students Likert questions about their prior experience with six tools/techniques in the course's technology stack. Post-course, students were asked the same six questions about their current experience. Due to the ordinal nature of the questions, the N = 49 paired student Likert responses are analyzed using non-parametric approaches. The six pre/post course experience questions are summarized using the median and interquartile range (IQR), which are visualized via a bar chart in Figure 1. These values are numerically shown in Table 2, which also includes results from quantifying the differences between paired pre/post course responses. These include Rosenthal's r for the effect size and a one-tailed Wilcoxon signed rank test p-value. As can be seen from Figure 1 and Table 2, students perceived the course improved their experience with Python, Git/Github, Docker, and unit testing. Students did not feel that the course significantly improved their command line skills, which is understandable since students often prefer using VS Code instead of command line. In addition to the six pre/post questions, eight post-course summary Likert questions were asked of the students. The Likert scale breakdown for each of these questions is shown in Figure 2. These results suggest students benefited from the "from scratch" and test-driven development of data science algorithms, though a large portion of the students would rather have directly used the libraries instead. All students agreed (or were neutral) that the course was valuable for their education.

The survey also asked the students to rank PA2-7 and the project based on which one they believed they learned the most from (PA1 was omitted due to its purpose as a small "warm-up" assignment). The students rated the assignments in the following order, from highest to lowest: PA6, PA5, PA4, Project, PA7, PA2, PA3 (see Table 1 for more information about the

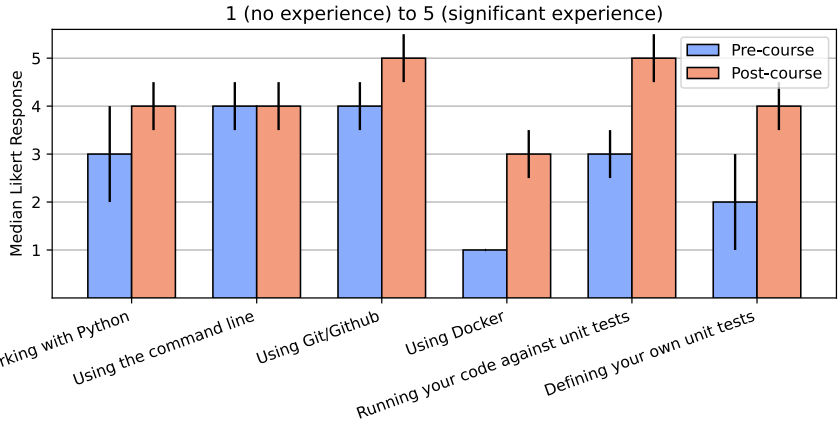


Figure 1: Median and interquartile ranges for student responses (N = 49).

Table 2: Summary of student pre-course and post-course paired Likert responses (N = 49). * = $p < 0.01$, ** = $p < 0.0017$ (Bonferroni-corrected)

Likert question	**Pre-course**	**Post-course**	**Effect size** r	**Wilcoxon test** p
Working with Python.	Median = 3 IQR = 2	Median = 4 IQR = 1	0.55	0.0000**
Using the command line.	Median = 4 IQR = 1	Median = 4 IQR = 1	0.20	0.0139
Using Git/Github.	Median = 4 IQR = 1	Median = 5 IQR = 1	0.26	0.0022*
Using Docker.	Median = 1 IQR = 0	Median = 3 IQR = 1	0.60	0.0000**
Running your code against unit tests.	Median = 3 IQR = 1	Median = 5 IQR = 1	0.50	0.0000**
Defining your own unit tests.	Median = 2 IQR = 2	Median = 4 IQR = 1	0.56	0.0000**

PAs). Students likely perceived they learned the most from PA6 because the TDIDT algorithm implemented in PA6 uses recursion, which is a topic students often do not have much practice with. Lastly, students were asked for any comments regarding the test-driven development, data science from scratch, or APIs/model deployment aspects of the course. One student commented,

"I think letting us build an algorithm by scratch was amazing and it helped me learn what is really going on behind the scenes in these python libraries." Another student commented, "It was nice going over unit tests and APIs in this class. I've had interviews where these topics have been covered, so it was nice to have it fresh in my mind + a little bit of practice."

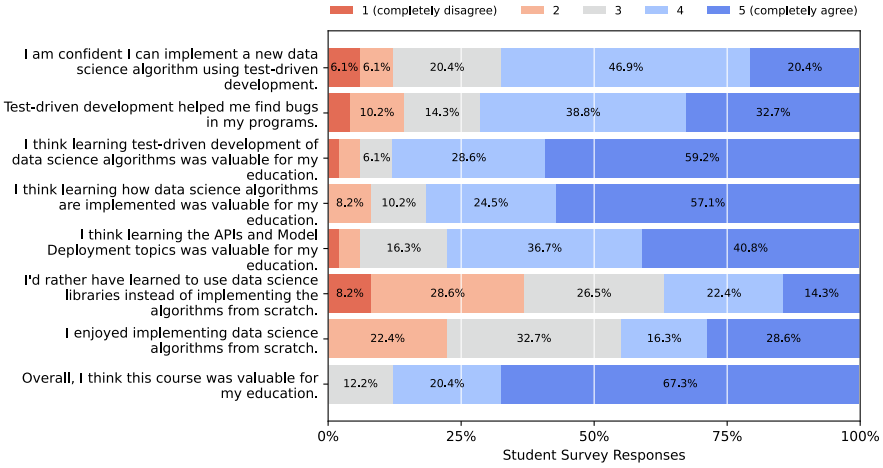


Figure 2: Student responses to post-course Likert questions (N = 49).

5 Summary and Future Work

This paper presents a course, CPSC 322 Data Science Algorithms, that helps prepare students for a career in data science by teaching popular Python library APIs and algorithm implementation details. The course has students use test-driven development to implement fundamental algorithms in their own `mysklearn` package with minimal use of non-standard libraries, while adhering to industry-standard library APIs. Students gain experience understanding data science algorithm execution and explaining output, while also sharpening their software engineering and data science tech stack skills. Future work aims to expand the algorithms students implement in the `mysklearn` package and to incorporate methods from the interpretable machine learning community.

Acknowledgement

The author wishes to thank Shawn Bowers for initial course design.

References

[1] Max Bramer. *Principles of Data Mining*. Springer, New York, NY, 3rd ed. 2016 edition edition, November 2016.

[2] Lito Perez Cruz. When Data Science Becomes Software Engineering:. In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 226–232, Funchal, Madeira, Portugal, 2017. SCITEPRESS - Science and Technology Publications.

[3] Boris Delibašić, Milan Vukićević, Miloš Jovanović, and Milija Suknović. White-Box or Black-Box Decision Tree Algorithms: Which to Use in Education? *IEEE Transactions on Education*, 56(3):287–291, August 2013.

[4] Brent Dykes. *Effective Data Storytelling: How to Drive Change with Data, Narrative and Visuals*. John Wiley & Sons, December 2019.

[5] Joel Grus. *Data Science from Scratch: First Principles with Python*. O'Reilly Media, Sebastopol, CA, 1st edition edition, April 2015.

[6] Johanna S. Hardin and Nicholas J. Horton. Ensuring That Mathematics is Relevant in a World of Data Science. *Notices of the American Mathematical Society*, 64(09):986–990, October 2017.

[7] Matthew Kirk. *Thoughtful Machine Learning with Python*. O'Reilly Media, Inc., 2017.

[8] Josua Krause, Adam Perer, and Kenney Ng. Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 5686–5697, New York, NY, USA, May 2016. Association for Computing Machinery.

[9] Sean Kross and Philip J. Guo. Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, Glasgow Scotland Uk, May 2019. ACM.

[10] Christoph Molnar. *Interpretable Machine Learning*. Leanpub, February 2018.

[11] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019. Number: 5 Publisher: Nature Publishing Group.

Appendix

The following code excerpt is an example of PA6 starter code for the implementation a decision tree classifier in a student's `mysklearn` package. Docstring comments have been removed for brevity.

```
# mysklearn/myclassifiers.py
import mysklearn.myutils as myutils

class MyDecisionTreeClassifier:
        def __init__(self):
                pass # TODO: fix this

        def fit(self, X_train, y_train):
                pass # TODO: fix this

        def predict(self, X_test):
                return [] # TODO: fix this

        def print_decision_rules(self,
                attribute_names=None, class_name="class"):
                pass # TODO: fix this

        def visualize_tree(self, dot_fname, pdf_fname,
                attribute_names=None):
                pass # TODO: (BONUS) fix this
```

The following code excerpt is an example of PA6 unit test starter code for test-driven development of the above `mysklearn/MyDecisionTreeClassifier` class' `fit()` and `predict()` methods. Docstring comments have been removed for brevity.

```
# test_myclassifiers.py
from mysklearn.myclassifiers import MyDecisionTreeClassifier

def test_decision_tree_classifier_fit():
        assert False == True # TODO: fix this

def test_decision_tree_classifier_predict():
        assert False == True # TODO: fix this
```

Researching Expensive Software Bugs: A Writing Assignment and Activity for Computing Students*

Tammy VanDeGrift
Computer Science
University of Portland
Portland, OR 97203
vandegri@up.edu

Abstract

This paper describes a writing and reflection activity about expensive software bugs. The assignment was designed with goals of building personal curiosity, resiliency, and responsibility, while creating connections to real world computing systems. In the reflection activity, students reported changes in their perspectives. Some of the reported updated perspectives were high-level – responsibility of software engineers in terms of safety, the impact of bugs on people and society – and some other lessons were more practical software development practices – take time to test and debug code, write good comments, ask others to review work, and expect the worst from users.

1 Introduction and Related Work

The Computing Curriculum 2020 report provides a framework for computing degrees and competencies [3]. Competencies combine knowledge, skills, and disposition to contextualize tasks. Computing dispositions encompass attitudes, habits, and social-emotional awareness and are as follows: proactive, self-directed, passionate, purpose-driven, professional, responsible, adaptable, collaborative, responsive, and meticulous. CS faculty must then create learning environments that build not only skills, but promote growth in computing

dispositions. In many undergraduate institutions, the educational mission of "head, hands, and heart" align with this framework. Knowledge is the "head", skills are the "hands", and disposition is the "heart". This paper reports on a writing assignment and in-class activity related to the professional and responsible dispositions.

Writing assignments provide opportunities for engaging curiosity, improving communication skills, and synthesizing knowledge. Research papers are common assessment tools in many disciplines, including computer science [1, 11]. Zobel describes effective CS research strategies and academic writing style for graduate students [14]. Writing-to-learn and concept mapping help students understand relationships [13]. Weikle describes a white paper and annotated bibliography assignments for algorithms [12]. There are several examples of writing assignments in computer science education for software documentation, lab reports, and research papers [1, 2, 4, 5, 7, 11]. This paper adds to the literature about writing assignments in computer science education. In particular, the paper describes a writing assignment about software bugs that could be used in many computing courses. The paper is organized into the following sections: a description of the assignment and gallery walk, data collected from students, results from the data, and conclusions.

2 Context for Expensive Software Bugs Gallery Walk

The writing assignment was designed for a Data Structures course at the University of Portland, a private, Catholic primarily undergraduate institution. The Data Structures course serves as the prerequisite course to all upper-division CS courses; additionally, it is a required course for CS majors, Electrical Engineering majors, and CS minors.

The university is part of the Kern Entrepreneurial Engineering Network (KEEN) whose mission is to instill undergraduates with mindsets of curiosity, creating connections, and creating value through opportunities [6]. Faculty design, assess, and share course activities at Engineering Unleashed [10]. In addition to curiosity and connections, this writing assignment addresses the CC2020 dispositions of being professional and being responsible. The activity includes an individual research/writing assignment and an in-class reflection; more details are below.

2.1 Part 1: Research paper (work completed individually outside class)

Students researched and wrote about an expensive bug using the prompt below:

How expensive are bugs? Not the kind that are alive, but the kind that are introduced through software. For this assignment, research a software bug.

Write a 500 – 600 word summary about the software bug (these will be shared with the class after the assignment is due). As you do your research, you may wish to think about these questions, but note that your paper should not just be an outline of answers:

1. What type of software contained the bug?
2. What company/organization produced the buggy software?
3. What programming language(s) and operating systems were used?
4. How was the bug found (public, internal testers, developers. . .)?
5. How expensive was the bug? You could define expense as dollars, employee time, effect on users, effect on larger system, management time
6. What factors contributed to the buggy software?

Specific formatting instructions, electronic submission instructions, and a grading rubric were provided; paper submissions were graded by the instructor.

2.2 Part 2: Gallery walk (in class – 25 minutes)

Once submitted, the instructor grouped papers into common themes. In the spring 2019 offering, the papers grouped into these themes: aircraft, video games, Y2K, space and missiles, Mariner 1, medicine, and other. Each theme had 2 to 6 papers. Prior to the gallery walk class session, the instructor created table tents with these themes.

A gallery walk is a pedagogical activity that gets students to physically move to different stations [9]. Usually, small groups of students rotate together to reflect and respond to materials at each station, similar to walking through an art gallery [9]. Because papers are difficult to read in groups (due to print size on paper limitations), the gallery walk for this activity was done individually. Each station was a represented by a single group-work table in the classroom.

Two days after the paper deadline, 25 minutes of a class session was used for the gallery walk. Students brought a printed copy of their paper to class (alternatively, the instructor could print copies of all papers) and placed their paper on the appropriate table station. Post-it notes and pens were available at each table, and students asked to make a comment or ask a question about the paper using the post-it notes. Students were encouraged to read a paper from five different tables (∼5 min per station) and read more if time allowed. After 25 minutes, students collected their own papers and read the comments and questions on the post-it notes. Note that this gallery walk was "quiet" since students read and reflected on their own; instructors may wish to do the gallery walk with student teams and make multiple copies of the same paper for ease of reading. Students did not earn formal homework credit for the gallery walk and reflection; instead, these counted toward the attendance portion of the grade.

2.3 Part 3: Reflection (in class – 5 minutes)

After reading through the post-it notes, students completed and submitted a short reflection with these questions:

After reading about software bugs that your classmates researched, please reflect and answer the following questions.

1. Describe at least two software bugs that you learned about today:
2. Doing my own research on expensive bugs and/or learning about bugs has changed my perspective of being an engineer or computer scientist in the following way(s):
3. Things that surprised me from reading other bug reports (if anything):

3 Research Study

Participants: Students completed all parts of the activity in Spring 2019. The expensive bug research paper was assigned in Spring 2020, and the gallery walk was scheduled for two weeks after students departed campus due to the covid-19 pandemic. Courses moved online in mid-March 2020 and became more asynchronous; therefore, the gallery walk was cancelled in Spring 2020. In hindsight, the gallery walk and reflection could have been moved online through shared documents, but the instructor chose to minimize "extra" activities for the remainder of the semester. Therefore, much of this paper reports on the results from Spring 2019 (non-pandemic semester). The participants included 26 students in Spring 2019 and 29 students in Spring 2020. The major composition in Spring 2019 was: 14 electrical engineering, 10 computer science, 1 business, 1 math. The major composition in Spring 2020 was: 10 electrical engineering, 10 computer science, 4 math, 2 mechanical engineering, 2 undeclared, and 1 business.

Data and Methods: In addition to the assigned work (research papers, post-it notes, and reflections), students completed an optional end-of-semester IRB-approved paper survey about course outcomes related to KEEN goals. This survey was conducted about a month after the gallery walk and questions addressed the entire course (not just the expensive bug activity). The end-of-semester surveys were administered and collected by a faculty member in the School of Education, and the completed surveys were provided to the instructor after final grade submission. The survey included quantitative questions (1=not at all, 2=very little, 3=to a small extent, 4=to a moderate extent, and 5=to a great extent) and open-text questions, as follows:

1. To what extent has your ability to identify links between course knowledge and real world systems increased during this class? (rate 1 to 5)

2. Describe a specific example of how your ability to identify links between course knowledge and real world systems increased in this class. (space for free text)
3. To what extent has your ability to understand ramifications of design decisions increased during this class? (rate 1 to 5)
4. Describe a specific example of how your ability to understand ramifications of design decisions increased in this class. (space for free text)
5. To what extent has your ability to recognize and explore gaps in knowledge increased during this class? (rate 1 to 5)
6. Describe a specific example of how your ability to recognize and explore gaps in knowledge increased in this class. (space for free text)

Table 1 shows the data collected from the activity submissions and from the end-of-semester survey in Spring 2019. A few students did not complete some parts of the activity, which is noted in Table 1. In Spring 2020, 26 of 29 completed the research paper and no students completed the other parts of the activity due to the pandemic, as explained earlier. The main data analysis method was content analysis of text [8].

Table 1: Number of activity items and survey submissions from Spring 2019

Item	**N (number collected)**
Student research papers (1 student did not submit)	25 of 26
Post-it notes from gallery walk (3 students absent, 1 did not submit)	15 of 22 papers had post-its 7 of 22 papers had no post-its
Reflection after gallery walk	23 of 26
End-of-semester KEEN survey	23 of 26

4 Results

The topics of the 51 research paper submissions formed 12 clusters. The most popular topics for software bugs were rockets and spacecraft. The list below shows the topics, ordered by most to least frequent:

- Rockets/Space, [16], *(e.g. Ariane 5, Mariner I, Mars climate orbiter)*
- Worms, [5], *(e.g. Morris, Stuxnet)*
- Medicine, [5], *(e.g. radiation – Panama, Therac-25, pacemaker)*
- Military, [5], *(e.g. Patriot missile, USS Yorktown, Dahran missile)*
- FDIV Intel floating point bug, [4]
- Video games, [4], *(e.g. Steam script, WoW Zulgurub, Corrupted Blood, Melee)*

- Infrastructure, [3], *(e.g. Soviet gas pipeline, airline baggage system, AT&T cell)*
- Y2K, [3]
- Aircraft, [3], *(e.g. F-35 fighter jet, Korean Air 801, Boeing 787 Dreamliners)*
- Stock exchange, [1]
- Apple imessage, [1]
- Webservers, [1]

Twenty-three students attended the gallery walk class session (one student did not write a paper). The post-it notes of the 22 papers were collected after the gallery walk. Of the 22, there was an average of 2.32 post-it notes per paper. The distribution was 1 with 7, 1 with 6, 1 with 5, 4 with 4, 2 with 3, 5 with 2, 1 with 1, and 7 with 0. Post-its fell into these themes: "this is scary", questions about specific bugs, questions about the aftermath, and positive comments about the quality of the paper. Students chose which topics/stations they wanted to visit; most papers were read by at least one other student. In future semesters, the instructor may wish to have students choose papers with no post-it notes when moving to the second or third station, so that all papers get some feedback.

Analysis of the students' reflections show that they took time to read and comprehend the papers; all 23 reflections had specific information about two different bugs in the answers to the first question. Responses to questions 2 (changed perspective) and 3 (surprising) were coded into themes [8]. The themes and frequencies are listed in Table 2; several themes were common across both questions in terms of impact and how small bugs often are. In a few cases, a single student response was coded into multiple, distinct themes, so the total number of themes per question exceeds 23. For example, *"It makes me realize that even the best mess up. All you can do is be sure to take the extra time to check your work."* was coded into two distinct themes: Even reputable organizations make mistakes (and) Take time to check work and test it. As a second example, the response *"Quality assurance and testing is so important because it can not only affect the functionality, it can also damage the businesses, families/community, and may even cause a generation-wide hysteria."* was coded into two themes: Take time to check work and test it (and) Impact, damage and cost of bugs.

The end-of-semester survey was completed by 23 of 26 students. Recall that the survey included questions about learning outcomes for the entire course. The course content included the C programming language (∼4 weeks) and data structures (linked lists, stacks, queues, trees, graphs, hash tables, sorting, big-O analysis) (∼10 weeks). The expensive bug writing assignment and gallery walk comprised one of many deliverables in the course. Students submitted

Table 2: Themes about students' changed perspectives and what was surprising (N=23)

Theme of student response	**perspective**	**surprising**
Take time to check work and test it	15	
Bugs are often simple/small and can cause major problems	6	8
Impact, damage and cost of bugs ($ and human life)	3	11
Risk/responsibility of software engineers	3	
Even reputable organizations make mistakes	2	2
Work with others	2	
Update software as hardware changes	1	
Do not release buggy code	1	
Expect the unexpected from users	1	
Communication is key	1	
Do not skimp on bits for storage	1	
Fix code when bugs are found	1	
Pay for high-quality software	1	
Comment with intention	1	
Lost funding		1
Pressure to re-use old code due to cost		1
Many different programming languages		1
One person wrote code without testing it		1
Bugs went unfixed for so long		1
Common mistakes across software		1
Almost experienced bug personally		1

weekly in-class labs to practice programming data structures, seven homework assignments (larger programs than labs), three midterm exams, and one final exam. With multiple assessments in the course, it was somewhat surprising that the research paper was stated by 14 different students as evidence for one of the three open-ended questions: identify links between course knowledge and real world systems (6 students), understand ramification of design decisions (8 students, 1 repeated for real world systems), and ability to recognize and explore gaps in knowledge (1 student). The research paper and gallery walk were a memorable learning activity. Here are some example end-of-semester survey responses about the expensive bug research paper:

- We did a lot of debugging stuff along with a small research paper, that exposed us to a lot of real world scenarios about bugs
- We had to research a programming bug that effected (sic) some machine. Learning about the bug actually gave me insight into what parts of a digital system are used for what and how software updates are utilized.
- We always discuss real-world applications of concepts such as stacks, queues, and trees. We wrote a paper identifying and researching a real-world, costly software bug which was very interesting. Out of all my classes, data structures is the most applicable to real world situations.
- The bug project allowed us to see the impacts of design decisions (primarily negative, but some positive)
- learning about the cost of computer bugs gave me perspective of the ramifications of bad designs
- We did a research summary on famous bugs that were either financially costs (sic) and/or fatal.

5 Discussion, Limitations, and Conclusions

Students remembered the expensive bug paper at the end of the semester, so it had lasting impact. The assignment was developed to help students see that most software is complex and debugging is an essential process. Students learn Java in the introductory programming courses and learn C very quickly in the Data Structures course. This assignment was a good fit since these students were familiar with two programming languages and could better understand bugs related to memory, overflow, and limited storage space. Being resilient through making mistakes and understanding the ethical responsibility of software engineers are part of the larger computer science curriculum; this assignment helps students to see that even the "best" engineers make mistakes and the importance of integrity and ethical conduct.

The instructor was interested to see how students define "expensive" for software bugs. Students defined expensive as lost revenue, as lost reputation,

as lost lives, and lost or damaged equipment. Hopefully, this helps them think more broadly about real-world systems, design decisions, financial decisions, and human safety.

This assignment may be used in any computer science course, since the learning objectives are not tied to specific course topics. It was heartening to see that the lessons students learned when researching expensive bugs included practical advice for themselves: allocating time for testing, importance of good commenting and communication, and having multiple people reviewing the same code.

The main limitation of this study is the participant pool – a small number of students from a single institution. The writing assignment was used in two semesters, but the data about the gallery walk, reflection, and end-of-semester survey were collected from just one cohort. Even so, the preliminary results indicate that students remembered the research paper, learned lessons from doing research and reading other students' work, and had some higher-level conclusions about their professional responsibility regarding safety and for checking their work. One student's reflection stated, *"Was surprised how many bugs were found, and all the good details provided. This is cool homework!! Interesting. So surprised by how expensive a bug is - a simple error can cost that much."* A student saying a homework assignment is "cool" is a big endorsement for learning.

The assignment details can be accessed at Engineering Unleashed (https://engineeringunleashed.com/card/1004) [10]. As more faculty across multiple institutions use this assignment, they are encouraged to study similar learning outcomes. The writing assignment can be expanded to focus on expensive engineering failures for more general engineering courses. Students showed curiosity when doing research and reading other students' papers. Students made connections by reflecting on how their perspectives changed. Hopefully, students' grew in the computing dispositions of responsibility and professionalism.

Acknowledgements

Thanks to Drs. Heather Dillon, Tim Doughty, Joseph Hoffbeck, and Nicole Ralston for help with preparing the survey, assistance with IRB, reviewing the assignment, and leading faculty development workshops. This work was supported by a Kern Entrepreneurial Engineering Network (KEEN) faculty grant and a Shiley faculty grant at the University of Portland.

References

[1] Robert F Dugan Jr and Virginia G Polanski. Writing for computer science: A taxonomy of writing tasks and general advice. *Journal of Computing Sciences in Colleges*, 21(6):191–203, 2006.

[2] Harriet J Fell, Viera K Proulx, and John Casey. Writing across the computer science curriculum. *ACM SIGCSE Bulletin*, 28(1):204–209, 1996.

[3] Association for Computing Machinery and IEEE Computer Society. Computing curricula 2020 (cc2020): Paradigms for global computing education. `https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf`.

[4] Mark E Hoffman. An updated taxonomy of writing in computer science education. *The Journal of Computing Sciences in Colleges*, page 175, 2011.

[5] David G Kay. Computer scientists can teach writing: an upper division course for computer science majors. In *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, pages 117–120, 1998.

[6] The Kern Entrepreneurial Engineering Network. The entrepreneurial mindset. `https://engineeringunleashed.com/mindset`.

[7] Dean Sanders. Writing activities can improve learning in computer science courses. *Computer Science Education*, 2(2):171–181, 1991.

[8] Steve Stemler. An overview of content analysis. *Practical assessment, research, and evaluation*, 7(1):17, 2000.

[9] The Teacher Toolkit. Gallery walk. `https://www.theteachertoolkit.com/index.php/tool/gallery-walk`.

[10] Engineering Unleashed. Welcome to engineering unleashed. `https://engineeringunleashed.com`.

[11] Tammy VanDeGrift. Coupling pair programming and writing: learning about students' perceptions and processes. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 2–6, 2004.

[12] Dee Weikle. Two concrete examples of upper-level writing assignments in an algorithms course. *Journal of Computing Sciences in Colleges*, 28(3):14–20, 2013.

[13] Ye Xiong and Yi-Fang Brook Wu. Write-and-learn: promoting meaningful learning through concept map-based formative feedback on writing assignments. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 552–553, 2017.

[14] Justin Zobel. *Writing for computer science*, volume 8. Springer, 2004.

Enhancing Cybersecurity Education and Workforce Through Colorado-Washington Security Scholar Program*

Yan Bai[1], Sang-Yoon Chang[2], Ken Lew[1,2], and Simeon Wuthier[2]
[1]School of Engineering and Technology
University of Washington Tacoma
Tacoma, WA 98402
[2]Computer Science Department
University of Colorado Colorado Springs
Colorado Springs, CO 80918

Abstract

Colorado-Washington Security Scholars Program (CWSSP) is a scholarship program for training and educating cybersecurity engineering students. Hosted in two universities for the students in the cybersecurity degree programs, the cross-campus program emphasizes virtual teamwork and collaborations in learning cybersecurity and executing the cybersecurity projects. This paper explains how the CWSSP program uniquely enhances the cybersecurity education and workforce development particularly focusing on the mechanisms to incorporate collaborations for the student scholars' training and the outcomes of the collaborations. We share our experience and insights from delivering the scholarship program in this paper.

1 Introduction

The CyberCorps (R): Scholarship for Service (SFS) program has a history dated back to 1998 where a directive was signed stating that the Executive Branch would need to assess the vulnerabilities of cybersecurity within the nation's critical infrastructure and to produce detailed plans in defending and protecting against future cyber threats. Hence in the year of 2000, SFS was created with the purpose in mind to further enhance the existing relationship between higher education institutions with different federal agencies to strengthen the Information Assurance (IA) disciplines as well as other federal initiatives within IA. Since then, SFS represents the opportunity for scholarship recipients to participate in Federal, State, Local or Tribal Government as a cybersecurity professional [1].

Started in August 1, 2019, with the same spirit of collaboration to strengthen the nation's cybersecurity professionals through SFS, the University of Colorado at Colorado Springs (UCCS) and University of Washington Tacoma (UW Tacoma) proposed a collaborative program called Colorado-Washington Security Scholars Program (CWSSP). The collaboration between the two universities stemmed from a grant awarded by the National Science Foundation under the CyberCorps (R): Scholarship for Service (SFS) program to establish the CWSSP program. Throughout this paper, we would explore the events and activities that took place between two campuses as well as some key data and outcome from CWSSP collaboration [6, 5].

2 Collaboration in research projects

With SFS fund support, UCCS and UW Tacoma have collaborated in cybersecurity research through a virtual team course, CWSSP conferences, and online collaborative platforms.

Virtual Teams Course - CWSSP emphasizes teamwork building on the cross-campus program being hosted across two geographically distant universities in Colorado and Washington states. To promote and facilitate virtual teamwork, CWSSP includes a Virtual Teams Course which teaches the participating student scholars the knowledge and techniques to improve on their virtual teams and collaboration skills and practice. The Virtual Teams Course is a hybrid course where the course kicks off from an in-person setting and is followed by the virtual/remote classes for the rest. A professor in communications delivers this unique course and designed it based on the state-of-the-art research in communications in social science, including the application of the transactive memory systems theory for a course assignment [9].

CWSSP Conferences (Student Presentations) - CWSSP Conference

is held once a year in one of the university campuses. The main purpose of this conference is to give scholars a platform to present and share their research with fellow scholars as well as other experts within the two campuses to further research collaborations between the two universities.

Below are a few examples of our collaboration projects:

Interdisciplinary Research - We initiated a large interdisciplinary research project among two institutions, UCCS and UW Tacoma in blockchain and supply chain. Our team consists of 15 members with various academic backgrounds including computer science, information systems, mathematics, business, and economics. Part of preliminary results from our SFS student members were presented at the 24th University of Washington Annual Undergraduate Research Symposium.

Blockchain Network Simulator to Enable and Facilitate Research – CWSSP program research resulted in a proof-of-work (PoW) blockchain network simulator [8] which has built-in random number generators (including that based on hash computations as used in real-world PoW blockchains). It can control the networking and consensus parameters for simulating different blockchain setups and environments, generate and log simulator-experimental data. The simulator reflects a UCCS scholars' understandings and experience working on an active Bitcoin node using the Bitcoin Core source code. When another UW Tacoma student scholar visited UCCS, the UCCS scholar provided a demo for the simulator to facilitate the simulator use for research. The simulator has been further discussed in virtual remote meetings to other scholars interested in blockchain research. We are actively distributing the simulator via open-source in Github so that it enables and facilitates others conducting research in blockchain networking security; the UCCS student leading the simulator development is in active communications with others interested in using the simulator for their own research, including a student working on his Thesis Dissertation outside of the US.

Blockchain Research Projects - Thanks to the CWSSP workshops, conferences, and offline meetings, there have been numerous opportunities to exchange research ideas and collaborate on research projects. As a result, CWSSP scholars and faculty have been productive with their research outcomes [9, 8, 4, 7, 2], by collaborating on and developing the necessary building blocks to assist in research directions. Highlighting such research outcomes include the aforementioned proof-of-work blockchain network simulator [8] and the following:

- ***Machine-learning-based anomaly detection on Bitcoin networking***: A CWSSP scholar implemented an active Bitcoin node on the Mainnet and built a networking sensor to collect the networking data. Based on such data, we built a machine-learning-based anomaly detection [4]

against the networking threats including those from the state-of-the-art research such as Eclipse attack [3] and denial-of-service (DoS).

- ***The discovery of denial-of-service vulnerabilities within the Bitcoin Core consensus protocol***: We discovered that the ban-score mechanism used in Bitcoin Core can provide vulnerabilities for DoS on blockchain via defamation, resulting in a node banning another legitimate node connection [2]. Such vulnerability has been disclosed and communicated to the Bitcoin Core developer team.

3 Constructing the Master's-to-PhD pipeline to generate the cybersecurity leaders with PhD degrees to secure the national cyberspace.

In addition to research collaborations, CWSSP also provides cybersecurity workforce training by educating cybersecurity leaders with PhD degrees. UW Tacoma and UCCS established an articulation agreement in 2017 to facilitate admission and degree completion of students earning Master of Cybersecurity & Leadership (MCL) degree at UW Tacoma to the PhD in Engineering-Concentration in Security degree program at UCCS. Per the agreement, all students that graduate from UW Tacoma MCL degree program with a 3.3 GPA or higher will be admitted directly into the PhD in Engineering-Concentration in Security degree program at UCCS, so long as they otherwise meet the UCCS requirements for admission. Additionally, 21 semester credit hours will transfer to with approval of UCCS Computer Science Department PhD in Security committee. This agreement has proven beneficial to both institutions. Seven MCL alumnus are pursuing PhD in Security at UCCS.

Apart from the seven MCL graduates that are pursuing the PhD program at UCCS, this year would be the first year where our MCL graduate student who is also a SFS scholar is entering the PhD program at UCCS in 2021 Fall. This MCL graduate student has participated in various research projects within the University of Washington system as well as collaborating with faculty and other PhD students at UCCS in research projects. With a program like CWSSP, that particular graduate student benefited from the tight relationship between the two campuses and was able to build a strong foundation in collaboration and in establishing research direction even before MCL graduation and into the PhD program. In other words, CWSSP is a platform for SFS scholars to network and work on research topics with other faculties and professionals in the area of cybersecurity. This enhances the overall experience of SFS scholars within CWSSP regardless if their next pathway would be joining the workforce or if they would continue on their education path within cybersecurity.

4 Project Outcomes

CWSSP was launched in Fall 2019 and is ongoing. This section describes the project outcomes in the first two years of the program since its launch. In these two years, CWSSP recruited and supported one PhD scholar, six master's scholars, and six bachelor student scholars.

Employment data - CWSSP is a part of the CyberCorps Scholarship for Service (SFS) program where the scholars are required to fulfill the service requirements to work in the US government after graduation. CWSSP provided a total of 9 scholar graduates and those graduates began their security-clearance-required employment at the US government, including Department of Defense (DoD), Department of State (DoS), National Renewable Energy Laboratory (NREL), Pacific Northwest National Laboratory, Washington State Government and National Security Agency (NSA).

Diversity - CWSSP prioritizes diversity and inclusion for the scholar selections. The students supported in the past two years include three female students, one Hispanic student, one African American student, and two Southeast Asian students.

Outreach - CWSSP is designed for outreach and specifically has the CWSSP Conference to outreach and network beyond the CWSSP and the participating departments. Our CWSSP Conferences in 2020 and 2021 included cybersecurity experts' presentations from the academic, government, and industry sectors. The scholars also attend the yearly nationwide SFS Career Fair events to represent CWSSP and network with the cybersecurity experts and potential employers beyond CWSSP.

Testimony - CWSSP has a systematic evaluation plan to solicit and receive feedback from the scholars to improve the program every year. The evaluations include semesterly surveys and individual communications between the scholar and the faculty. The student responses have been overall positive about the program and demonstrates that the CWSSP program improves the student's aptitude in cybersecurity research/project and virtual teamwork. Example qualitative responses about the overall program quality and usefulness include: "It [the CWSSP program] is potentially life changing and people need to know about the opportunity" and "This program has provided me a great opportunity to achieve more than just an education" and "It met and exceeded expectations." Other feedback/responses helped improve the CWSSP events and components, for example, "I think a research discussion session would be a nice touch, since everyone's research is similar, everyone would have things to contribute" and "[...] I enjoyed being involved in events as a group where I could interact with others."

5 Lessons learned

From Faculty - CWSSP is a unique CyberCorps SFS program enhancing the host institutions' degree programs by focusing on cybersecurity research/projects and virtual teamwork/collaborations. Having these focuses anchor the design and the execution of CWSSP components and mechanisms provided a unifying theme and goal. Sharing these visions and the success cases also helped engage the student scholars.

From Students - Additionally, the program has been beneficial to the participants through the hands-on research projects and collaboration across campuses. In fact, the two student authors of this paper have utilized many CWSSP opportunities for networking and gaining real-world experience. By driving the research across many projects, and learning effective collaboration skills, this has been a valuable learning experience not only for the education provided by the program, but from the interactions with potential future employers, and experts in the cybersecurity field.

6 Conclusions

The CWSSP collaboration is unique and it shows the creativity of the collaboration through various events and projects while all communication was through a virtual environment. According to our project outcomes, not only does CWSSP enhance the cybersecurity education between the two campuses, but the program also better helps and prepares its scholars for the cybersecurity workforce within government agencies. More importantly, it broadens the perspective of all its scholars with multidisciplinary research and increases their professional network by connecting with other professionals outside of their own campus.

7 Acknowledgements

The research was in part supported by the National Science Foundation (NSF) Grants 1921576 and 19922410.

References

[1] CyberCorps. Cybercorps: Scholarship for service. https://www.sfs.opm.gov.

[2] Wenjun Fan, Hsiang-Jen Hong, Simeon Wuthier, Xiaobo Zhou, Yan Bai, and Sang-Yoon Chang. Security analyses of misbehavior tracking in bitcoin network. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE, 2021.

[3] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.

[4] Jinoh Kim, Makiya Nakashima, Wenjun Fan, Simeon Wuthier, Xiaobo Zhou, Ikkyun Kim, and Sang-Yoon Chang. Anomaly detection based on traffic monitoring for secure blockchain networking. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.

[5] UW Tacoma. Cybercorps scholarships to expand cybersecurity education. https://www.tacoma.uw.edu/news/article/cybercorps-scholarships-expand-cybersecurity-education.

[6] Colorado Springs University of Colorado. Colorado-washington security scholars program. https://cwssp.uccs.edu.

[7] Li-e Wang, Shan Lin, Yan Bai, Sang-Yoon Chang, Xianxian Li, and Peng Liu. A privacy preserving method for publishing set-valued data and its correlative social network. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[8] Simeon Wuthier and Sang-Yoon Chang. Proof-of-work network simulator for blockchain and cryptocurrency research. In *ICDCS 2021 - 41st IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021.

[9] Kay Yoon and Sang-Yoon Chang. Teaching team collaboration in cybersecurity: A case study from the transactive memory systems perspective. In *2021 IEEE Global Engineering Education Conference (EDUCON)*, pages 841–845. IEEE, 2021.

Is Programming Relevant to CS1 Students' Interests?*

Kevin Buffardi and Subhed Chavan
Computer Science Department
California State University, Chico
Chico, CA 95929-0410
kbuffardi@csuchico.edu, sachavan@mail.csuchico.edu

Abstract

By surveying students (n=198) at the beginning of an introductory programming (CS1) course, we polled students' hobbies and interests and investigated which they considered relevant to Computer Science (CS) or coding. Analysis of student responses used Grounded Theory and revealed that the most popular interests were games, athletics, music/-film/audio/visuals, STEM, and nature/outdoors. Of those most popular interests, students only widely perceived games and STEM as hobbies that could be relevant. Hypothesis testing using chi-square goodness of fit revealed that gender minorities were disproportionately less likely to be interested in games and STEM than their male counterparts. However, no disparities in interests were found for racial minorities.

1 Introduction

Relating course materials to students' interests may promote motivation and engagement, which is essential in improving learning as well as retention in the major. Meanwhile, Computer Science (CS) course enrollments have disproportionately represented male White and Asian students so there are ongoing efforts to broaden participation, especially to improve gender and racial representation. To promote engagement in introductory programming (CS1)

courses, educators have adopted different themes and applications such as incorporating robots, games, or media computation. However, to best suit CS1 courses for improved student motivation, we first must understand students' interests.

In an exploratory study, we investigated students' hobbies and interests as they began CS1. To gauge whether or not students' considered coding and CS as a discipline relevant to their interests, we identified a primary research question: **RQ1. What popular interests do students perceive as relevant to CS/coding?** Conversely, we also set out to discover **RQ2. What popular interests do students perceive as *not* relevant to CS/coding?**

Since CS1 classes are often predominantly male and White or Asian, we did not want the interests to only represent those backgrounds. Accordingly, we investigated **RQ3. Are there differences in how interested different genders and races are in common CS1 applications?**. Finally, with a motivation for broadening participation in CS, we explored **RQ4. Are there interests that uniquely appeal to gender and racial minorities?** With these four research questions in mind, our study examined: *is programming relevant to CS1 students' interests?*

2 Background

Constructivism [7] is a theory of cognitive development that describes learning new knowledge by integrating it with preexisting knowledge. Relating new information to what students already know should help them construct new skills and understandings. Additionally, students who feel motivated also tend to demonstrate more engagement, longer persistence, and better learning outcomes [8]. In an effort to motivate students in CS1 courses, educators have explored different approaches to introduce the discipline in a fun and relevant manner.

Developing games in programming classes is a popular application that has yielded improved student engagement [5]. Another common approach is to apply introductory coding skills to operate robots [9] or even digital representations, such as Karel the Robot [1]. Alternatively, *Media Computation* [2] courses leverage coding to manipulate digital representations of sound, images, movies, and even produce music [3]. While studies of each of these applications have found positive outcomes, there is an open question whether they have a neutral, positive, or negative affect on diversity in CS.

For decades, CS has lacked diversity. The problem has been so persistent that the National Academies of Sciences, Engineering, and Medicine has called for innovating practices to drive interest in CS [6]. In addition, the National Center for Women & Information Technology's framework for engaging women

in CS encourages making course content matter to students with practices including "Use Meaningful and Relevant Content" and "Make Interdisciplinary Connections to CS". Accordingly, this paper explores differences in interests between gender and racial minorities and how they relate to CS.

3 Method

During the first week of Fall 2020 and Spring 2021 semesters at California State University-Chico (Chico State), CS1 students completed a survey (n=198) on their identities and interests. The survey included the following items *(with form entry methods specified)*:

- What is your major? *(Free response)*
- Gender *(Select one from: Female, Male, Non-binary, Prefer not to say, or Other [with free response])*
- Race (select all that apply) *(American Indian or Alaska Native, Asian, Black or African American, Hispanic or Latin American, Native Hawaiian and Pacific Islander, White, or Other [with free response])*
- What are your hobbies and interests? (separate by commas) *(Free response)*;
- Of those hobbies and interests, which do you believe can be relevant to Computer Science or programming? *(Free response)*.

The online survey did not require items to be answered, but most students responded to every item.

The CS1 course at Chico State serves as the first in a three-course sequence focused on programming and algorithms, all taught in C++. Both Computer Science and Computer Information Systems majors require the three-course sequence while students from several other disciplines take CS1 to fulfill electives in their respective majors or for general education. Consequently, we coded Computer Science and Computer Information Systems as *Majors* and all other responses as *Non-Majors*.

Students' self-reported gender included 154 Male (78%), 41 Female (21%), 1 Non-binary, 1 Prefer not to say, and 1 no response (<1% each). Since enrollment has been predominantly male in recent decades, we coded males as *Gender Majority*, Female and Non-binary as *Gender Minority*, and excluded *Prefer not to say (and omissions)* for analysis on gender.

Similarly, recent enrollment has also skewed White and Asian so those responses were coded as *Racial Majority* and all others were coded as *Racial*

Minority. Students who selected multiple races were only counted as *Racial Majority* when White *and* Asian were the *only* selections. When one or more of any other race was selected, we coded the mixed identity as *Racial Minority*. Student responses indicated 119 Racial Majority (60%) and 77 Racial Minority (39%) students; two students omitted a response and were excluded from analysis on race. Chico State is an official Hispanic Serving Institution, lending to higher representation in CS1 courses than national averages. However, Hispanic student enrollment in CS1 is still not proportionate to the University's demographics.

We manually reviewed students' responses that indicated their interests and which of those interests could be relevant to Computer Science or coding. Using Grounded Theory [4], we coded responses into fifteen common categories of interests: games; reading; arts and crafts; music, film, and audio/video; athletics; Science, Technology, Engineering, and Math (STEM); other academic disciplines (non-STEM); socializing; sleep and relaxation; animals; culinary; fashion and cosmetics; nature and outdoors; travel; and automotive.

We also inspected students' responses about which of their interests were relevant and coded each of their interests as either *Relevant* or *Not Relevant*. Table 1 summarizes students' interests and their perceptions of whether or not those interests can be relevant to Computer Science or coding.

Table 1: Students' Interests and Perceived Relevance

Interest	Relevant	Not Relevant	% Relevant
STEM	60	1	98%
Games	99	20	83%
Other Academic Disciplines	14	6	70%
Automotive	10	10	50%
Fashion and Cosmetics	2	3	40%
Arts and Crafts	14	23	38%
Reading	9	21	30%
Music, Film, and Audio/Visual	20	48	29%
Sleep and Relaxing	2	5	29%
Socializing	4	22	15%
Athletics	13	94	12%
Nature and Outdoors	6	43	12%
Culinary	1	13	7%
Travel	0	8	0%
Animals	0	3	0%

4 Results

The most popular categories for students' interests were: games (n=119, 60%, such as playing video or tabletop games); athletics (n=107, 54% such as team sports or exercising); music, film, and audio/visuals (n=68, 34%, such as playing musical instruments or watching movies); STEM (n=61, 31%, such as tinkering with electronics or learning about science); and nature and outdoors (n=49, 25%, such as hiking, camping, and gardening). Fewer than a quarter of the students identified each of the other interests.

Most students perceived their interests in games (83%) and STEM (98%) as relevant to CS. However, the other most popular interests were mostly perceived as not relevant. Only 12% of students interested in athletics perceived it as relevant. Music, film, and audio/visual were only perceived as relevant to 29% of students with those interests while only 12% of students who enjoy nature and outdoors considered their interest relevant.

To answer **RQ1, games and STEM are popular interests that students perceive as relevant to CS**. It is fairly common to find CS courses that integrate coding with these applications, as we discussed in the Background section. In investigating **RQ2, we found that interests in athletics; music, film, and audio/visuals; and nature and the outdoors were popular but considered not relevant to CS**. Media Computation courses may reinforce relevance to students' interests in music, film, and audio/visuals. However, there may be untapped potential in incorporating applications of athletics or nature and the outdoors in coding courses.

We also investigated whether there are differences in interests between White and Asian males who account for the majority of CS majors and historically underrepresented groups. Table 2 summarizes the percentage of students with each interest, as dissected by gender and racial majority/minority.

To investigate common applications in CS1 courses, we examined the interests: games; STEM; and music/film/audio/visual. To determine whether interest was proportionate to gender and race representation within the courses, we performed chi-square goodness of fit tests. Goodness of fit tests whether the observed proportions of majority and minority students match the expected proportions, given the class demographics (summarized in the Methods section).

When investigating interest in games, we found that most of students in the gender majority (n=104, 68%) expressed interest, while most of the gender minority (n=15, 36%) did not. Interest was disproportionate (χ^2=5.50, df=1, p<.05) with statistical significance. However, interest was proportionate (χ^2=0.50, df=1, p=.48) between racial majority (n=76, 64%) and minority (n=43, 56%), due to a lack of statistical significance. **Games garnered disproportionate interest from male students at the sake of gender**

Table 2: Students' Interests by Demographic

Interest	Gender		Race			
	Majority	Minority	Majority	Minority	Major	Non-Major
Games	68%	36%	64%	56%	67%	56%
Reading	12%	29%	14%	17%	15%	15%
Arts/Crafts	14%	36%	18%	21%	14%	22%
Music/Film	33%	38%	31%	39%	36%	34%
Athletics	56%	48%	57%	51%	55%	54%
STEM	36%	14%	35%	25%	29%	32%
Other Disc.	8%	17%	11%	9%	8%	12%
Socializing	12%	19%	12%	16%	12%	14%
Sleep/Relax	3%	5%	2%	6%	3%	4%
Animals	1%	5%	2%	1%	1%	2%
Culinary	3%	21%	5%	10%	6%	8%
Fashion/Cos.	1%	7%	3%	3%	3%	3%
Nature/Out.	25%	24%	29%	18%	22%	27%
Travel	5%	2%	4%	4%	3%	5%
Automotive	12%	2%	10%	9%	8%	12%

minorities, but interest in games was proportionate between racial groups.

We compared STEM interests and found that gender majority students (n=55, 36%) were disproportionately more likely to be interested (χ^2=4.87, df=1, p<.05) than gender minority students (n=6, 14%). Interest was approximately proportionate (χ^2=1.69, df=1, p=.19) between racial majority (n=42, 35%) and minority (n=19, 25%) students. Similarly to games, **male students reported disproportionately higher interest in STEM hobbies than gender minorities, but there was no significant variation in interest between racial majority and minority students**.

We also compared interest in music, film, and audio/visuals. The chi-square goodness of fit indicated no significant variation from proportionate interest between gender majority (n=51, 33%) and minority (n=16, 38%) (χ^2=0.24, df=1, p=.62) nor between racial majority (n=37, 31%) and minority (n=30, 39%) students (χ^2=0.85, df=1, p=.36). As a result, our investigation of **RQ3 indicated that gender minorities have disproportionately less interest in both games and STEM hobbies, while there were no significant differences between racial majority and minority students**. **Interests in Music, film, and audio/visual were proportionate for both racial and gender minorities**.

Finally, for **RQ4** we inspected gender and racial minorities' interests. Ath-

letics was the most popular interest among gender minorities. While gender minorities did not express as much interest in games as male students, they did express more interest in music/film/audio/visual and were more interested in arts/crafts than their male counterparts. Racial minorities also expressed proportionate interest in athletics, music/film/audio/visual, and arts/crafts. It may also be noteworthy that both gender and racial minorities indicated more interest than their majority counterparts in several other categories including: reading, socializing, culinary, and sleeping/relaxing.

5 Discussion and Conclusions

In this study, we investigated students' interests and whether they thought those personal interests could be relevant to Computer Science (CS) or coding. A survey of CS1 students (n=198) indicated that games and STEM hobbies were popular and widely considered relevant to CS. However, athletics, music/film/audio/visual, nature/outdoors, and arts/crafts were popular interests that most students perceived as *not relevant* to CS or coding. To broaden CS' appeal to students with different interests, CS1 courses (and any preceding CS curricula) may incorporate applications to these areas.

The question of whether or not any hobby can be related to CS/coding may even draw differences of opinions between CS academics and professionals. However, with a focus on discovering engaging course material that appeals to students' interests, any hobby that can incorporate data and an algorithm to process it has potential to motivate students. Novel applications that are relevant to students' interests, but not often represented in CS courses, may broaden participation and promote engagement.

Many CS1 courses adopt game or robot applications for coding, but our study found that gender minorities are disproportionately less interested in those hobbies than male counterparts. However, we found no such disparity between music/film/audio/visual hobbies based on either race or gender. We also discovered popular interests that are usually untapped in CS1, including athletics, nature/outdoors, and arts/crafts. Curricula that concentrate on game and STEM applications of coding may be harmful to gender minorities in the absence of options to apply coding to other interests. Curricular and course development should consider these findings in an effort to broaden participation in CS.

Nevertheless, future work is necessary to address limitations and build upon this study. Firstly, the study was conducted at a Hispanic Serving Institution so similar phenomena might be found at similar institutions with comparable demographics. However, research into students' interests and perceptions of CS should replicate this study in different settings. Additionally, students

surveyed were all enrolled in a CS1 course so our findings may be impacted by selection bias; future work should also investigate interests and perceptions of those students who have not (yet) enrolled in any CS courses.

Pairing this study's survey with a corresponding post-semester survey could reveal whether or not CS1 course interventions have an impact on students' perceptions. Moreover, students have many characteristics that shape their identities and interests. This study should only be interpreted as a preliminary investigation to discovering opportunities to broaden the appeal of CS, not to diminish students to individual characteristics. Future work should also investigate how intersections of identity and background may influence students' interest and perceptions in CS.

6 Acknowledgements

The authors of this paper are grateful for the instructors who incorporated the surveys into their courses, which was essential for this work.

This material is based upon work supported by the Learning Lab, an initiative of California Governor's Office of Planning and Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Learning Lab.

References

[1] Byron Weber Becker. Teaching cs1 with karel the robot in java. *SIGCSE Bull.*, 33(1):50–54, February 2001.

[2] Mark Guzdial. A media computation course for non-majors. *SIGCSE Bull.*, 35(3):104–108, June 2003.

[3] Brian Magerko, Jason Freeman, Tom Mcklin, Mike Reilly, Elise Livingston, Scott Mccoid, and Andrea Crews-Brown. Earsketch: A steam-based approach for underrepresented populations in high school computer science education. *ACM Trans. Comput. Educ.*, 16(4), September 2016.

[4] Patricia Yancey Martin and Barry A. Turner. Grounded theory and organizational research. *The Journal of Applied Behavioral Science*, 22(2):141–157, 1986.

[5] Briana B. Morrison and Jon A. Preston. Engagement: Gaming throughout the curriculum. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, page 342–346, New York, NY, USA, 2009. Association for Computing Machinery.

[6] Engineering National Academies of Sciences and Medicine. *Cultivating Interest and Competencies in Computing: Authentic Experiences and Design Factors*. The National Academies Press, Washington, DC, 2021.

[7] Jean Piaget. A history of psychology in autobiography. 4:237–256, 1952.

[8] P.R. Pintrich. A motivational science perspective on the role of student motivation in learning and teaching contexts. *Journal of Educational Psychology*, 95(4):667–686, 2003.

[9] Jay Summet, Deepak Kumar, Keith O'Hara, Daniel Walker, Lijun Ni, Doug Blank, and Tucker Balch. Personalizing cs1 with robots. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, page 433–437, New York, NY, USA, 2009. Association for Computing Machinery.

Cybersecurity Virtual Labs Open-Source Teaching Initiative: Creating an Entrepreneurial Mindset – Curiosity, Connections & Creating Value (3Cs)*

Radana Dvorak[1] *and John L. Whiteman*[2]
[1]*City University of Seattle*
Seattle, WA 98121
dvorakradana@CityU.edu
[2]*Intel*
bulepdx@gmail.com

Abstract

Cybersecurity classes continue to present challenging problems to computer science departments. IT departments often don't have the resources to set up specialized labs to support the curriculum and purchasing third-party cyber labs are not an option for many departments due to reduced budgets. Setting up environments is often left to the instructor. Instructors having to create labs is a problematic option since it is a very time-consuming. COVID-19 has recently compounded this problem due to universities having to close-down labs.

We discuss delivering a cybersecurity curriculum designed based on a framework that introduces a topic, followed by publicized case study and then real-life hands-on practice lab using current industry applications. The modules are designed to help students develop entrepreneurial mindset [13, 7, 8] based on the three C's (Curiosity, Connections and Creating Value for society). The framework is part of an open source project initiative that allows universities, students, and others to contribute their lab work to a public repository hosted by an entity like GitHub. We believe the success of this project has great potential for community colleges and universities.

1 Introduction

This paper reports two years of teaching a cybersecurity curricula using the Keen framework with the goals of engaging and exciting students about the cybersecurity field to address the job shortage, and also address the challenging problems computer science departments undergo to prepare and set up specialized labs to support the cybersecurity curriculum.

Opportunities for engineering student to develop an entrepreneurial mindset has been given attention in the last 5 years [13, 7, 8]. The teaching model presented is motivated by the Kern Entrepreneurial Engineering Network framework (KEEN) which has three key elements [13]:

- Curiosity -Understanding and engaging on a topical issue to "understand the broader world, look toward the future and explore multiple perspectives."
- Connections – connecting the issue and solutions to the real world. "Think outside the box; place old ideas in the new context and gain insights"
- Creating Value – the lab environment teaching students to think through the problem, identify new solutions to "seek opportunity, understand the stakeholders and have impact."

The paper first reports the success of delivering a cybersecurity curriculum designed based on a framework consisting of introducing a topic, a case study, followed by real-life application practice lab. We also describe the platform developed to support the framework. These labs were deployed outside the university IT infrastructure and successfully supported hands-on-practice, and practical experience industry expects of graduates [5].

In teaching cybersecurity classes for the last two years, we found that students learn best by doing hands-on exercises immediately after a security concept is introduced in the lecture. The more traditional lecture followed by a different day for labs, often with only TAs present, has shown to be less effective. Furthermore, the authors found that the more realistic the scenarios, the more students became curious and engaged. This interest the topics and experiences generated connections with internships, attending cybersecurity talks and led to a significant number of students making decisions to pursue cybersecurity internships, apply for cybersecurity jobs or apply to Master's programs in cybersecurity. This initiative continues to be work in progress as we are expanding and updating the labs. The outcomes reported are only from University wide end of semester course evaluations; these evaluations are standardized across all courses. Empirical research to evaluate the outcomes was not carried out. This work will be carried out in the second phase of this work.

The realistic scenarios also provided students with the confidence to discuss real-life cases and demonstrate their knowledge and practical know-how in technical interviews when applying for internships and jobs. Documented complaints from the industry report that students have the 'theoretical' knowledge and can talk about how to approach the problem; however, when it came to actual demonstration of their skills, they were not competent and generally lacked hands-on experience. Computer Science departments, world-wide, have not prepared students to meet the demands of industry. In August 2018, the Forbes Technical Council published an article, "The Cybersecurity Talent Gap Is An Industry Crisis", stating 3.5 M jobs world-wide will not be filled by 2021 [10]. A recent workforce study published in 2019 by (ISC)2 [12], the world's largest non-profit association that certifies cybersecurity professionals, reported that in order to meet the needs of cybersecurity workforce globally, skilled cybersecurity professionals needs to grow by 145% to meet the demand. The report further stated that in order to meet the crucial needs of American businesses, the US cybersecurity workforce needs to grow by 62%!

The lack of students' practical experience often stems from difficulties university departments have setting up environments, restricting students to practice skills such as pentesting[1], sniffing networks[2], or analyzing malware[3]- skills that are covered in cybersecurity courses.

We outline the solution to developing practical virtual labs based on real-world cybersecurity scenarios students will encounter at workplace and how it aligns with the 3C's framework and conclude by proposing interested parties, students, and faculty contribute to developing and adding cybersecurity lab activities to an open source collaborative infrastructure platform.

2 Curiosity: Understanding the issues, explore multiple perspectives and look to the future

Three cybersecurity classes were delivered in a classroom/lab setting before COVID-19 restrictions. One course was held in Spring 2020, half in the classroom, and the remainder online because the University closed in-person classes. In Fall 2020, the course was delivered entirely online. Having the infrastructure set up before the University classes were entirely online made the transition seamless for students and the instructor compared to what other engineering courses experienced.

[1]Pentesting (penetration testing), also referred to as ethical hacking is an 'authorized' cyberattack. Can be simulated or real-life. Pentesters are hired to find security vulnerabilities in systems.

[2]Network sniffing, also known as packet sniffing (or sniffers) are snapshot copies of data flowing over a network without altering it or redirecting it.

[3]Malware is software designed to damage, disrupt or gain access to a computer system.

Lectures were designed around specific topics, taking to more than 20-30 minutes each. A short discussion followed. The labs were created based on highly publicized security incidents, like data breaches. After the breaches were introduced, students discussed the incidents and shared their thoughts on the ethics of the organizations' actions and / or legal outcomes. Initial work started on introducing ethical components to cybersecurity modules in 2019 [4].

3 Connections: Thinking outside the box; gain insight; pursue knowledge and integrate it with students' own discoveries

The lab sections followed right after the discussion. The first lab is designed to teach students how to set up their own laptop environments. Although students worked individually, they were able to chat and share in an online public forum. We stressed the importance of figuring out the problem by themselves. If students requested help, instructors would first quiz the students on what was tried and guide them to discover the solution.

Here are several examples of the typical problem-solving scenario given to the students where each lab created simulation of a real-world cybersecurity event:

- Extracted and analyzed malware from a binary image using open source forensic tools. It was the infamous WannaCry ransomware that affected over 200,000 computers in 2017.
- Found a famous fugitive, John McAfee, by extracting coordinates from pictures taken of him while on the lam in Central America [11].
- Created an encryption and decryption C program for one assignment and have it continuously bombarded with garbage data to see if any security vulnerabilities exist. If so, students learn how to write more secure code to fix the problems.
- Ran a capture-the-flag event that simulated a vulnerable website that sold juice.
- Students participated in the National Cyber League (NCL) competition during the term. This competition is only for college students. Over 4,700 students participated.

4 Creating Value: Virtual Lab Infrastructure: As educators, we must train students to persistently anticipate and meet the needs of a changing world

Creating virtual labs for cybersecurity classes has been given attention in the last few years, and ASEE has published papers on the topic [9, 1, 16, 4]. Some universities create their labs while others use the NSF-funded SEED Labs Project [3, 2]. The drawbacks of instructors creating labs are addressed above. Relying on graduate students is hit or miss, often not sustainable due to graduate students' priorities to graduate, not to mention, no one to support what was initially developed. The solution to the ongoing challenges CS departments experience, is to use a virtual lab framework as designed and developed for our courses. We propose to share what we have started to develop and launch an open source initiative for the academic community.

5 Class Offering

The same course was offered over a two year period, and as the labs were continued to be added on to, we optimized the delivery for the students.

5.1 First Class Offering

Dedicated lab space on campus was not planned since the course was starting from scratch. Although the curriculum was prepared in advance and presented as such in the syllabus, the actual lab content was still in active development throughout entire course. Using a cloud-based hosting environment for the labs was initially considered, but the costs were not factored in on time as student fees. Initial budget estimates seemed high too. A localize lab environment needed to be created. Every student had a laptop, but their operating systems (OS) were not the same. Some students used macOS, others used Windows, and a small number of students preferred to work in a Linux environment. For those students who used the same OS, many ran different versions which caused installation issues due to compatibility issues with software. We had to standardize the lab environment starting with the OS. One way to accomplished this was to use virtualization technology. Students installed a special software application called a hypervisor on their OS. A hypervisor runs a virtual machine (VM). Think of a VM as a software application that when started acts like a fully functional computer running inside the host OS. Multiple VMs can run at the same time as long as the host system has enough processing and memory resources to handle it. Each VM has its own configurable CPU, memory, networking and disk space. Students did not have to purchase addi-

tional hardware and many hypervisor programs are free. We took advantage of this technology to install a common OS to host the lab environment. As long as students installed a compatible hypervisor, they could run the same VM regardless of what host OS they used. Applications ran consistently and troubleshooting issues became easier. Figure 1 shows a simplified virtualization configuration on a computer.

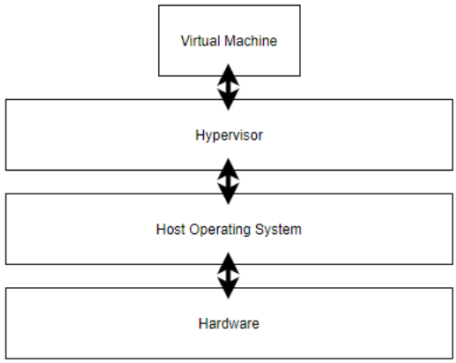


Figure 1: Virtualization

During the first week of class, we asked students to install the hypervisor software on their laptops. We assigned this exercise as *Lab 0*. We also had them create three VMs that were needed to support the future labs. Students learned the basics of virtualization during the lab. We also took the opportunity to teach them basic virtualization security. For example, we explained why some types of network connections are less secure than others. We also had students prepare special cryptographic keys that were later used to digitally sign their homework to prevent cheating. Students learned the mathematical behind these keys later during the cryptography sections of the class, thus giving them a real-world example of how they can be applied.

Students experienced difficulty with this lab because they were not accustomed to working exclusively in a command line environment. Most only used graphical user interface-based operating systems like macOS and Windows. The university offered a Unix Tools Laboratory course that students were expected to take as a prerequisite. Many challenges students faced during the lab are the same ones faced in real-world security jobs. To solve these issues, students learned to collaborate and communicate with others as the KEEN framework shows. To create a VM, students start by installing an OS. We used Metasploitable, Kali Linux and Ubuntu. Figure 2 shows how the hypervisor can manage multiple VMs on the same host.

Once the OSes were installed, students were asked to download files that

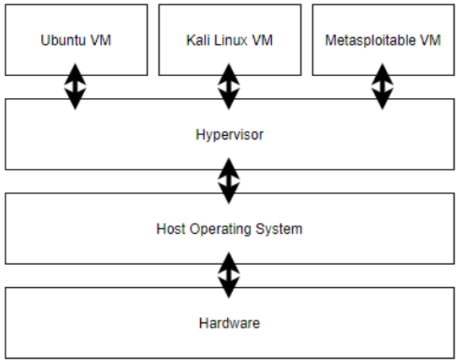


Figure 2: Multiple virtual machines

contained our lab materials from a public source code repository hosted on GitHub. The hosting is free. We provided helper scripts to automate the setup process for each VM. Figure 3 shows our first VM framework, including the applications, tools and samples that we used for our labs and class activities along with the network connections between the VMs and GitHub.

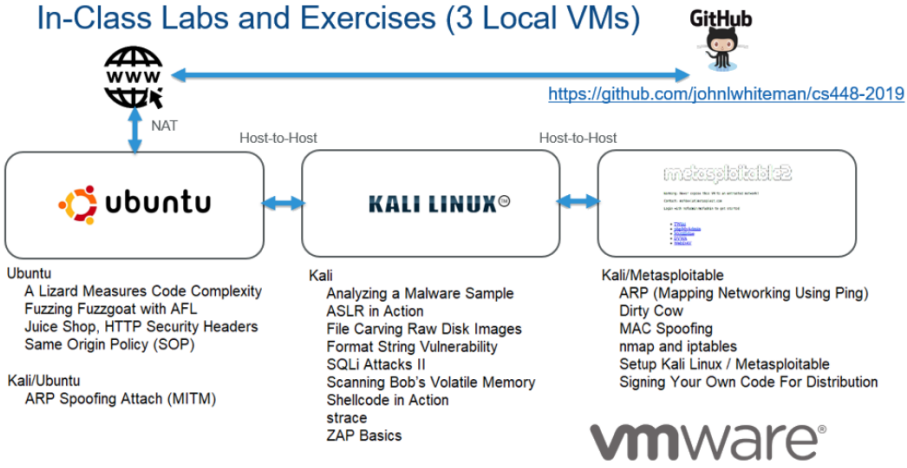


Figure 3: Our first offering

The entire process to setup took less than an hour to complete with most time spent on waiting for the programs to download and install. The class did not need to rely on IT department support. Another advantage of using VMs

is isolation from the host OS and institution's network. For example, one of our labs involved reverse engineering a famous malicious computer worm called Stuxnet. It can only be executed on the Windows operating systems. All of our VMs are Linux-based so there was no worry about the worm activating and spreading to the outside world through execution. Security professionals call this sandboxing and students got to apply this concept through engineering though and action.

Advantages

- Standardized and secure lab environments
- Automated setup and teardown
- Students took ownership of the labs, not dependent on the IT department
- Instructors taught students about basic VM security principles
- All software was free

Disadvantages

- Running multiple VMs concurrently caused less powerful laptops to perform poorly
- Storage size of all of VMs took significant disk space, up to 75 GB.
- Misconfiguration of network connects between VMs could disrupt the IT network
- Some students were not prepared to work in Linux command line environments

5.2 The Second Class Offering

We reduced the number of VMs to only one. We chose Kali Linux 2020 since it already came pre-packaged with many of the security tools used during the first iteration. As a result, the storage footprint size to less than 25 GB since we extraneous content was omitted. We initially coordinated with the IT department to host the VM on the school's network. Many advantages came from this approach. First, students could use the institution's computer resources instead of their laptops. The machines were powerful, and it also solved the potential problem for future students who did not own a laptop. Second, instructors were given administrative access to students' VMs in case they needed to remotely assistance with running the labs. Third, IT could effectively isolate the VM network from the rest school's main network to prevent accidental disruptions from running misconfigured security tools. Forth, once the instructor designs the VM, it can be handed-off the IT to mass deploy to the students' accounts.

Unfortunately, we experienced a few problems. First, when students had technical issues with accessing their VM instance, they needed to contact the IT department that could cause delays. Second, the lab content was always evolving which meant that IT needed to refresh the students' VMs for every new update provided by the instructor. After a valiant effort working with IT, we decided to revert to the local VM installation to take back control of the lab environments. Since the labs were refined to just one VM, the installation and maintenance effort went smoother.

We also introduced the use of containers. They too perform a kind of virtualization but from an application perspective. Each container can serve a special purpose. For example, we can use one container to house a web application that has known security vulnerabilities then another to run tools that scan the web application. What is beneficial about containers is that they already come prepackaged with all of the right dependencies installed. Students can focus on the learning objective at hand instead of worrying about installation. Figure 4 depicts what that might look like.

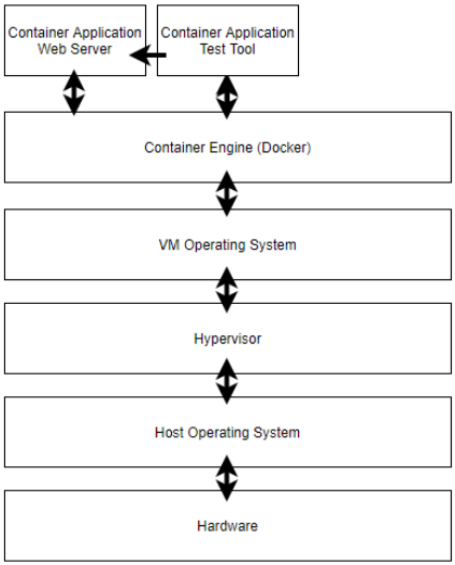


Figure 4: Containerization with VM

We wanted to keep each container as application specific as possible, since we can treat each one as an ingredient for a lab recipe. For example, other containers were created that perform a different kind of testing against the vulnerable web application. Figure 5 below depicts a multiple container frame-

work.

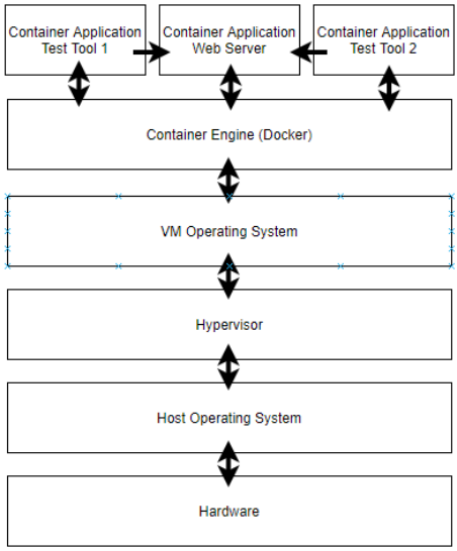


Figure 5: Multiple containers

Packaging individual labs into containers standardizes the lab environments even more. All students get the same recipe for each lab with each designed a specific learning purpose in mind. Consequently, we discovered that can add more labs and class activities than before with containers since everything is ready to go when students come the lectures. We did not have to rewrite each lab from scratch, we stitched containers together to create new labs as such shown in Figure 6.

Each lab can consist of one-to-many interconnected containers and because we are running inside a VM, none of these container connections leave the VM thus protecting the IT network.

- *Lab 1 is a container that connects to an intentionally vulnerable website called JuiceShop. The lab is a capture-the-flag (CTF) event that allows students to learn how to hack and, in doing so, collect points. They are free to use offensive tools such as fuzzers to find these vulnerabilities. Everything is done securely in containers inside the VM.*

- *Lab 475 is a container that connects to an intentionally vulnerable OS called Metasploitable. It's completely isolated from the host including its antimalware tools.*

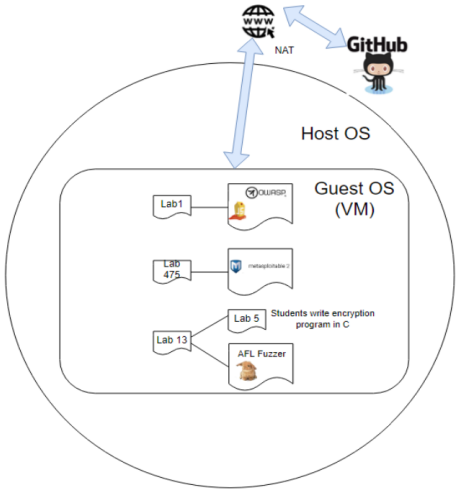


Figure 6: Container interconnectivity

- *Lab 13 is an example of a lab that combines two other labs. Lab 5 is an older lab completed by the students who were supposed to write a fully functional encryption and decryption program in the C language. Later in the term the instructor assigns this lab to see if the AFL fuzzing tool can find vulnerabilities in their program by continuously bombarding it with garbage data.*

5.3 The Latest Class Offering

Although we considered running the labs in a hosted cloud environment such as Amazon AWS or Microsoft Azure, the matter of funding was still in progress. We continued to add new labs and class activities levering on our existing virtualization and containerization approach. The culmination of these activities led to this extreme example of how multiple exercises can be stitched together to create a new and compelling assignment that uses real-world scenarios:

1. *Students are given a file containing recorded wireless traffic from an overseas hotel in Asia.*
2. *They are requested to analyze it using a few open source tools such as Wireshark and tcpreplay. The traffic is encrypted except at the start where a cryptographic handshake takes place to establish the secret key.*
3. *Students are asked questions about this exchange including the random values that are passed in the clear between the client and server. Later the*

students are prompted to find anomalous traffic. This is next to impossible since the traffic is encrypted, so the instructor helps by providing the actual secret key to decrypt it.

4. *The students quickly observe that a suspicious Windows executable is sent from the server to the client.*
5. *They use a forensics tool to extract it from the rest of the file and with a free malware analyzer, they find that it contains the infamous WannaCry ransomware worm that affected hundreds of thousands of computers across 150 countries in just days, including hospitals and other critical infrastructure. The damage could have been worse though except that a young British security researcher found a hidden "kill switch" hardcoded in the program to stop the virus from spreading.*
6. *The students download the Ghidra reverse engineering tool provided by the National Security Agency (NSA). They find the same kill switch as the security researcher with excitement because they are looking at the real thing.*
7. *Instructor delivers a lecture about legal and ethical issues associated with computer security. Instructor informs students that the security research was also wanted for alleged security crimes he committed in the past. All of this was happening at the same time.*
8. *Students are given an assignment to debate the issue and make a decision about criminal activities – court room scenario. Students need to decide whether the hacking hero should be exonerated of his alleged previous criminal past.*

5.4 Future Class Offering

We strongly believe using a cloud service provides the best experience for the students and have seen this from previous work at Oregon Health and Science University (OHSU) where a web class we created had its labs securely hosted by Amazon Web Services. The model is similar to the IT networking effort we described earlier at UP, but here instructors have administrative access to the dashboard to maintain and update the labs. Cost is the biggest factor though when considering using the cloud to host labs. Charges include what software is being installed, the hardware that it uses, how much data is transferred, how many instances per student and how long each remains active. These costs add up fast and can exceed any reasonable lab fee for students. We estimated hundreds of dollars per semester per student at regular prices. Still most cloud providers offer academic prices to universities. Our goal is to continue to pursue cloud hosting as the best option.

6 Creating an Open Source Community to Provide Free Content

Successful open source projects [14, 15] have a strong, active, transparent community behind them. Our goal is to create an open source project that leverages trusted and reliable content from anyone who wants to contribute. We start with a free public Git repository provided by companies such as like GitHub, GitLab or BitBucket.

6.1 Contributions

Any member in the open source community can contribute, whether new lab exercises, bug fixes, testing or documentation. New lab content is first peer-reviewed by the community to ensure that high quality and security best practices are followed before submission. If bugs are found, they can be reported and tracked in the repo's ticketing database. All labs require documentation using a well-known formatting language such as Markdown. Students can even contribute to the labs. They are the end-users who can add the most significant value to them. We strongly encourage this since contributions to an open source project focused on security are practical experiences students can showcase in their resumes. Our goal is to reach a wider academic community, excite them about what is being created, and contribute to a successful project. Since this is an open source project, it should be treated as using standards the community expects. At the heart of contribution is something called a Pull Request on GitHub as described next.

Members are not allowed to add their content without first going through a pull request (PR) process. Each PR contains a description of the proposed changes along with the content itself. Any member of the community can review the PR, often adding suggestions and comments along the way to ensure quality. Once a PR is thoroughly reviewed, a core project member can accept the PR by merging or rejecting it. The PR process is the same used by most open source projects that use Git. Community peer reviews help keep the integrity of the project intact as well as encourages collaboration and transparency.

6.2 Maintenance

Figure 7 proposes a well-known branch flow used by many other organizations. Starting from bottom to top, the new branch contains cutting-edge content not officially released. Once mature, it gets merged to the Dev (Development) branch, where it gets integrated with other new content. The Dev branch is eventually tagged with a release number then merged to the Release branch. How often a release occurs is determined by the community; either by quarter or

semester which can nicely align with academic calendars. Releases are merged into the Master branch. The Patch branch contains emergency fixes in case critical/high bugs are reported shortly after a release. They can be released quicker to fix serious issues. All other bugs are tracked in a bug database which are triaged and dispatched to developers to fix in the Dev branch.

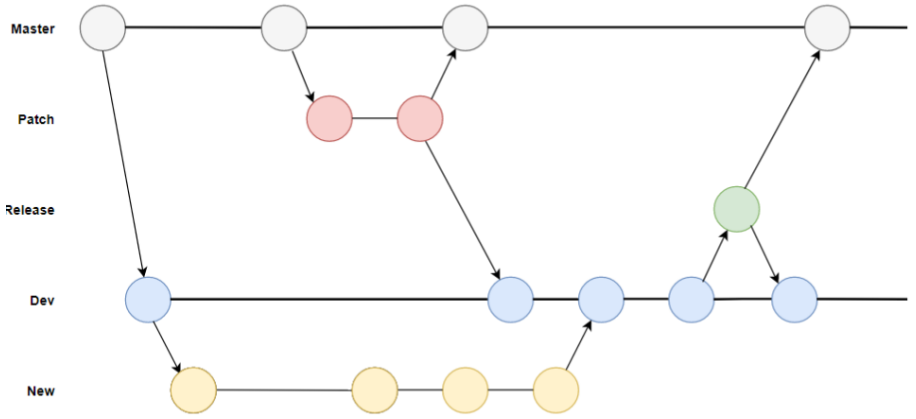


Figure 7: Git repository

7 Course Outcomes

No formal survey was designed and administered to evaluate the outcomes. The instructors used the University wide end of semester formal student course evaluation, compared grades from their previous offerings and the improved result of students participating in the National Cyber League (NCL).

Students were very engaged in the labs; they reported enjoying them and felt it was a valuable learning experience. Grades were higher than prior offerings of similar content, which was contributed to the engagement in being active participants in 'real-world' cybersecurity scenarios. As mentioned above, we did not carry out empirical research to report student performance and engagement. However, the end of the term evaluation showed that using real life scenarios helped students understand and apply their knowledge of cybersecurity concepts. Furthermore, students were given the opportunity to experience what a team in industry experiences when confronted with a breach.

A few students used the open response area to provide more general feedback on the course. Most of the responses were positive, with students offering appreciation for this assignments' complexity compared with the more traditional lab elements. Sample of students' comments:

- *I do think that in some cases it's good to get your hands dirty. I wish the course could be expanded or divided into more electives, as the material here is really important.*
- *Labs are fun and relatable to course content – homework is derived from lectures, which I like – the course gives great background on cybersecurity – I really enjoyed the extra credit assignments*
- *I liked the interactive lessons*
- *Each of the topics taught was very interesting and the labs were also just as interesting as well.*

8 Discussion

Based on students' engagement, improved grades and end of the course evaluations, the delivery model has been deemed successful. Both instructors will continue to teach using the open source lab framework. Of the students who completed the courses, twenty percent (20%) decided to pursue cybersecurity fields from our junior and senior standing students. Two students were admitted to the Carnegie Mellon University Cyber Security Master's program; other students were successful in securing internships. This is a success for the instructors dedicated to exciting students to pursue careers in cybersecurity to help fill the extreme technical gap currently experienced in industry. Recently reported by "The Cybersecurity Talent Gap Is An Industry Crisis", stating 3.5 M jobs world-wide will not be filled by 2021 [10]. A recent workforce study published in 2019 by (ISC)2 [12], the world's largest non-profit association that certifies cybersecurity professionals, reported that to meet the needs of the cybersecurity workforce globally, skilled cybersecurity professionals need to grow by 145% to meet the demand. The report further stated that to meet American businesses' crucial needs, the US cybersecurity workforce needs to grow by 62%!

9 Conclusion - future work

The success of open source projects is well documented [14, 15]. Most recently, GitHub, acquired by Microsoft Inc. 2018 launched GitHub Learning in 2019, fully supporting education lab classrooms [6]; however, there is still not one open source project specifically aimed at supporting educators to teach cybersecurity curriculum. Our work following the KEEN framework has demonstrated initial success with just several modules; future work will continue to build on the initial structure and ideally tested by partnering universities. Processes need to be in place to ensure quality and security. We are proposing and calling for an open source project initiative that allows universities, students, and others to contribute their lab work to a public repository hosted by an entity like GitHub. Our goal is to provide a centralized, trusted platform where tools

are well vetted and cataloged beforehand. As we continue to add to the labs, we plan to formalize our research and collect data on the success of the program, and research whether the outcomes led to continued interest in pursuing cybersecurity internship, jobs and enrolling in graduate programs in cybersecurity. The authors are also interested in collecting demographic data to study non-represented populations in cybersecurity. The research outcomes will be reported in future conferences/papers.

References

[1] Aaron Carpenter. A hardware security curriculum and its use for evaluation of student understanding of ece concepts. In *2018 ASEE Annual Conference & Exposition*, 2018.

[2] Wenliang Du. Seed labs. `https://seedsecuritylabs.org`.

[3] Wenliang Du. Seed labs: Using hands-on lab exercises for computer security education. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 704–704, 2015.

[4] Radana Dvorak, Heather Dillon, Nicole Ralston, and Jeffrey Matthew Welch. Exploring ethical hacking from multiple viewpoints. In *2020 ASEE Virtual Annual Conference Content Access*, 2020.

[5] Nuryake Fajaryati, Muhammad Akhyar, et al. The employability skills needed to face the demands of work in the future: systematic literature reviews. *Open Engineering*, 10(1):595–603, 2020.

[6] GitHub. Github learning labs. `https://lab.github.com`.

[7] Löbler. Fostering an entrepreneurial mindset by using a design. `https://journals.sagepub.com`.

[8] Rita Gunther McGrath and Ian C MacMillan. *The entrepreneurial mindset: Strategies for continuously creating opportunity in an age of uncertainty*, volume 284. Harvard Business Press, 2000.

[9] Alyssa Mendlein, Aunshul Rege, and Thuy-Trinh Nguyen. Cybersecurity awareness and training through a multidisciplinary osint course project. In *2020 ASEE Virtual Annual Conference Content Access*, 2020.

[10] Brian NeSmith. The cybersecurity talent gap is an industry crisis. *Forbes (online)*, pages 08–09, 2018.

[11] Daria Onitiu. 'the case for open source'a report on matt wells' seminar by ninso, northumbria internet & society research group. *Northumbria Internet & Society Research Group (June 19, 2020)*, 2020.

[12] (ISC)2 Cybersecurity Workforce Study. Strategies for building and growing strong cybersecurity teams. `https://www.isc2.org`.

[13] Engineering Unleashed. Keen - the framework. `https://engineeringunleashed.com/mindset-matters/framework.aspx`.

[14] Mike Volpi. How open-source software took over the world. `https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/`.

[15] Steven Weber. *The success of open source*. Harvard University Press, 2004.

[16] Austin Whipple, Keith B Smith, Dale C Rowe, and Samuel Moses. Building a vulnerability testing lab in an educational environment. In *2015 ASEE Annual Conference & Exposition*, pages 26–301, 2015.

Visual Sensor Networks: Analysis of Environmental Impacts via Computational Thinking*

Tisha Brown-Gaines
Computer Science
College of Sciences and Mathematics
Belmont University, Nashville, TN 37212
tisha.gaines@belmont.edu

Abstract

Visual Sensor Networks (VSNs) are comprised of camera nodes that are capable of acquiring, distributing, and processing images. Thus, providing rich information about a given event. However, image sensing is an extremely powerful electrical consuming mechanism, causing visual sensor networks to indirectly contribute to greenhouse gas emissions. As our society adopts emerging technologies central to the field of Internet of Things (IoTs) such as smart homes and smart surveillance systems, it is imperative to consider energy consumption and efficiency to create environmentally friendly technology. In this study, we will investigate the environmental footprint of visual sensors and develop a program that optimizes object tracking and energy efficiency via computational thinking. The energy consumption of a PixyCam2 will be measured with a multi-meter while performing several algorithms to analyze various metrics central to the visual sensor's functionality.

1 Introduction

Energy consumption is an important factor in creating sustainable and eco-friendly technology. Approximately forty percent of energy consumed by the

United States is used to generate electricity [7]. As a result, electricity use constitutes a significant portion of every individual's environmental footprint [7]. Cameras provide wide coverage and rich information, producing large amounts of data [5]. However, image sensing is extremely power consuming [3]. Thus, the usage of visual sensor networks can contribute to air pollution and greenhouse gas emissions. Using computational thinking techniques (CTT), we will decompose this complex real-world problem into manageable concepts for student groups to recognize patterns, use abstraction and construct feasible algorithms [9]. By reducing the energy consumed by visual sensor networks from non-renewable resources, such as fossil fuels, one can reduce greenhouse gas emissions, therefore, mitigating the effects of climate change.

2 Related Work

The energy consumption of VSNs have been studied under different coverage conditions and network states [1]. Singh et al. analyzed energy efficiency in wireless visual sensor networks under the conditions of barrier coverage, blanket coverage, and grid coverage [8]. Margi et al. studied VSNs executing various benchmarks such as processing, acquiring images and communicating as the sensors are in sleep or awake modes [6]. Casares et al. utilized peer-to-peer (P2P) object tracking and detection in wireless embedded smart-camera systems in their analysis of power consumption and performance to showcase the importance of lightweight algorithms and data transfer [3]. LiKamWa et al. focused on the issue of high-power consumption in CMOS image sensors [4]. Obringer et al. considered the environmental footprint of Internet use [7]. Although there exist research pertaining to visual sensor networks, there exists no research analyzing the environmental impact posed by visual sensor networks in relation to their energy consumption.

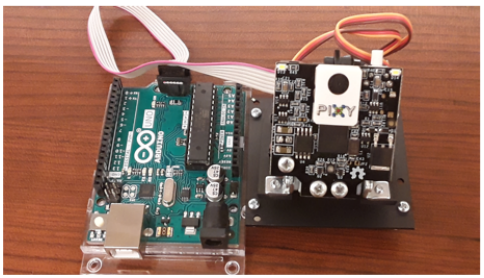


Figure 1: Micro-controller and PixyCam2 Setup

3 Problem Formulation

We will study the environmental footprint of visual sensor networks as it relates to object tracking and energy consumption. We then determine the extent to which a visual sensor contributes to one's environmental footprint by calculating the carbon emissions produced based on the energy consumed when performing various algorithms such as: object tracking utilizing pan-tilt mode, stationary object tracking and paused mode. We developed a sleep program that optimizes object tracking and energy efficiency. Additionally, this study focuses it's methodology on computational thinking to promote well-defined problem solving and research skill sets in secondary and undergraduate learners [9].

3.1 Energy Consumption Model

The PixyCam2 utilized is representative of a single camera node within a visual sensor network as shown in Figure 1. The camera node is capable of tracking objects that enter its field of view (FOV) as discussed in the recent study [2]. The Arduino Uno micro-controller was utilized to program algorithms that instruct the camera nodes to capture information pertaining to the object being tracked. As shown in Figure 2, a laptop is used to power the camera and micro-controller through a micro-USB and USB A/B cable, respectively. The camera and micro-controller can communicate through the 6-pin-to-10-pin-IDC cable.

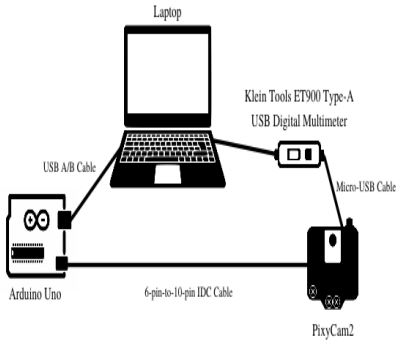


Figure 2: Model Configuration Setup

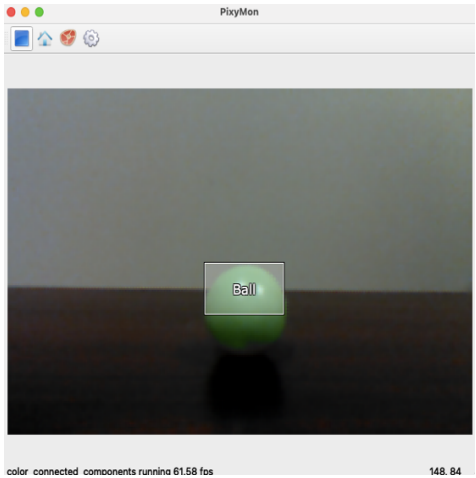


Figure 3: Object Tracking Environment

The Klein Tools ET900 Type-A USB Digital Multi-meter (DMM) is used to measure the voltage and current flow as certain algorithms are executed. The DMM is attached to the laptop while the camera is attached to the DMM.

3.2 Pan-Tilt with Object Tracking

The camera node makes use of the pan-tilt mode and color tracking capabilities as shown in Figure 3. The pan-tilt mode allows the camera to rotate left, right, up, and down. The PixyCam2 is programmed to track and follow an object based on its color, allowing it to track a small green plastic ball. The number of objects detected is returned and displayed on the serial monitor of the Arduino platform followed by the object's location in terms of xy coordinates and color.

Listing 1: Sleep mode coding example

```
void sleep() {
  if(delayRunning && ((millis()-delayStart) >= DELAY_TIME)
  && pixy.ccc.numBlocks == 0) {
  delayRunning = false;
  delay(5000);
  LowPower.powerDown(Sleep_8S, ADC_OFF, BOD_OFF);
  LowPower.powerDown(Sleep_2S, ADC_OFF, BOD_OFF);
  delayStart = 0;
    } }
```

3.3 Stationary with Object Tracking

The camera node is mounted on the pan-tilt stand but no longer utilizes that feature. However, it utilizes the color tracking feature to identify the green ball as long as it remains present within the node's field of view (FOV). The object's xy coordinates and color is displayed on the serial monitor as shown in Figure 4.

3.4 Pause Mode

The camera node is mounted on the pan-tilt stand but it is no longer connected to the Arduino board. The program that the PixyCam2 is running is stopped as the camera is put in paused mode. No object tracking or object information is obtained.

4 Sleep Algorithm

The sleep algorithm was developed with the aim of optimizing energy efficiency and object tracking. The camera node makes use of the pan-tilt mode and

color tracking features ensuring object tracking. In order to optimize energy efficiency, the Arduino board is put to sleep if no object is identified within 30 seconds. The Arduino board sleeps for 10 seconds and automatically wakes up, repeating these actions. The sleep and wake process is highlighted in the Listing 1: Sleep algorithm code snippet referenced. The Arduino microcontroller notifies the user whether it is awake or asleep using the serial monitor. The xy coordinates and color is displayed.

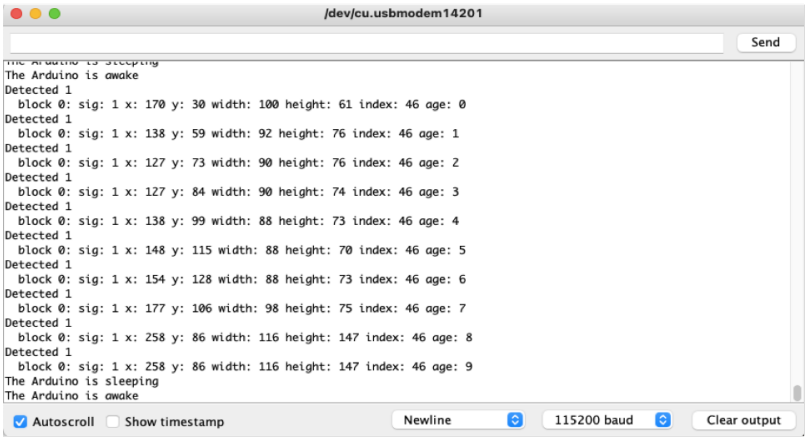


Figure 4: Sleep Algorithm Output

5 Methodology

The DMM provided the voltage and current of an algorithm in real-time. Both the voltage and current were collected within the time frame of 1 minute intervals for each algorithm. Figure 5 depicts the power consumption of each algorithm at a given time calculated by $P = V \times I$, where **P** is the power, **V** is the voltage, and **I** is the current; the power is converted and depicted in kilowatts. The average power consumption was then determined. Power is the rate at which energy is consumed and the energy consumed by each algorithm is given by the formula $E = P_{a_{vg}} \times T$, where **E** is energy. $P_{a_{vg}}$ is the average power and **T** is the execution time, which is 1 hour [3]. Figure 6 shows the CO_2 emissions produced by each algorithm. To determine the CO_2 emissions, the energy provided in kilowatts per hour (kWh) was converted to kilograms (kg) of CO_2 based on the conversion factor provided by the EPA: 0.000709 kg CO_2 /kWh [6]. $C = E \times F$, where **C** is the CO_2 emissions emitted, **E** is energy, and **F** is the conversion factor.

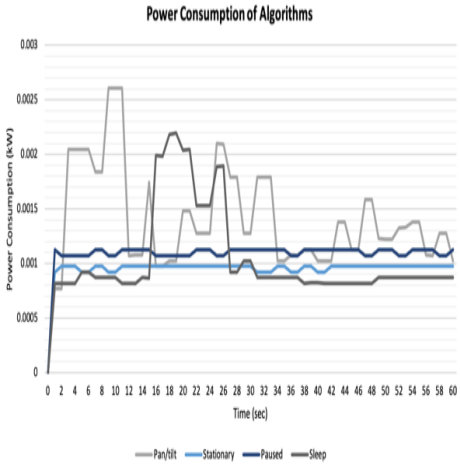


Figure 5: Performance Results (kW)

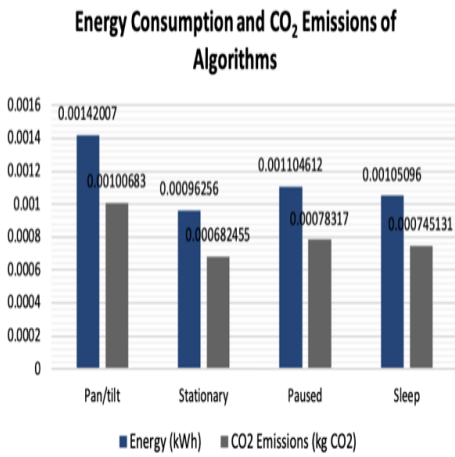


Figure 6: Performance Results (CO_2)

6 Performance Evaluation

Figure 6 shows the energy consumption and CO_2 emissions of the camera node for each algorithm. The results show that energy consumption is slightly disproportional to CO_2 emissions. Regarding the baseline algorithms, the pan-tilt algorithm consumed more kWh, thus emitting the most CO_2 emissions. The stationary algorithm produced the least amount of CO_2. The sleep algorithm was successful in optimizing energy efficiency and object tracking as it consumed less energy than the pan-tilt algorithm, thus producing less CO_2 while being able to track an object beyond the coverage of the stationary algorithm. The study of energy consumption and the environmental impact of visual sensor networks can be extended to consider different coverage models including: barrier, blanket and point to strategically save power by exploiting the redundancy of coverage in a multi-camera system.

7 Future Work

The data obtained in this study can serve as the foundation for calculating the energy consumption of larger-scale simulated environments and Industrial IoTs technologies. Additionally, a system scaling model highlighting the transition from a single camera node to multiple camera nodes is targeted. Given the positive student outcomes seen in grade improvement, student-instructor rapport and content engagement we hope to expand the study to integrate the computational thinking techniques into a broader curriculum. We noted that computational thinking techniques allow instructors to connect different topics

of study by reducing subject-matter lines to emphasize unifying concepts by implementing one well-defined step at a time. As computational thinking is referred to as a "21st Century skill set" plans to engage undergraduate cohorts into Big Idea Micro-Projects (BIMPs) that will be integrated into intro-level computing, data science and mathematics courses.

References

[1] Sidra Aslam, Farrah Farooq, and Shahzad Sarwar. Power consumption in wireless sensor networks. FIT '09, New York, NY, USA, 2009. Association for Computing Machinery.

[2] Tisha Brown, Zhonghui Wang, Tong Shan, Feng Wang, and Jianxia Xue. Obstacle-aware wireless video sensor network deployment for 3d indoor monitoring. pages 1–6, 12 2017.

[3] Mauricio Casares, Alvaro Pinto, Youlu Wang, and Senem Velipasalar. Power consumption and performance analysis of object tracking and event detection with wireless embedded smart cameras. pages 1 – 8, 10 2009.

[4] Robert LiKamWa, Bodhi Priyantha, Matthai Philipose, Lin Zhong, and Paramvir Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, page 69–82, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Sharad Malik and Andrew Wolfe. Power analysis of embedded software: First step towards software power minimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2:437 – 445, 01 1995.

[6] C.B. Margi, V. Petkov, K. Obraczka, and R. Manduchi. Characterizing energy consumption in a visual sensor network testbed. In *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, pages 8 pp.–339, 2006.

[7] Renee Obringer, Benjamin Rachunok, Debora Maia-Silva, Maryam Arbabzadeh, Roshanak Nateghi, and Kaveh Madani. The overlooked environmental footprint of increasing internet use. *Resources, Conservation and Recycling*, 167:105389, 04 2021.

[8] Juginder Singh, Manas Mishra, and Mohd Khan. Energy efficient video surveillance in wireless sensor networks under grid coverage over barrier coverage. *SSRN Electronic Journal*, 01 2018.

[9] Jeannette Wing. Computational thinking. *Communications of the ACM*, 49:33–35, 03 2006.

A Conceptual Framework for an Introductory Machine Learning Course*

Anthony D. Bowman and Leon Jololian
Electrical and Computer Engineering
University of Alabama at Birmingham
Birmingham, AL 35294
anbowman@uab.edu, leon@uab.edu

Abstract

Historically, computer science curricula have largely focused on the development of algorithmic solutions. However, in recent years a new paradigm has emerged which focuses on machine learning as a data-driven approach to problem solving. To better equip students with the background needed for this emerging method of problem solving, the curricula would benefit from including a greater emphasis on the concepts frequently found in a data-driven workflow. Furthermore, continued advances in hardware should be considered to exploit parallelism in computations. Other concepts from machine learning include data cleaning, feature engineering, and concurrency in computation. In this paper, we propose a conceptual framework for an introductory course in machine learning with a data-driven workflow. With this framework, students will be exposed to the high-level design and concepts of data-driven development, gradually equipping them with the tools to fully grasp the machine learning paradigm. Further research is ongoing to construct a development environment that supports this framework for full adoption of the machine learning paradigm.

1 Introduction

Since their inception in the latter half of the 20th century, computer science curricula have faced the challenge of preparing students for the demands placed on them by their post-collegiate career paths. For the better part of that history, the predominant method of problem solving involved the development of algorithmic solutions, therefore curricula were adopted with courses teaching concepts and skills which supported such development including traditional and more recent models for software development ([1, 9]). In recent years, new data-driven approaches like machine learning have emerged and continued to gain widespread prominence and complexity in both industry and academic research [6]. While there is significant overlap in the background knowledge required for both approaches, there are also some concepts in the data-driven workflow that bear greater emphasis because of the critical role they play in generating solutions. Thus, there is a need for curricula to adapt to the increasing demand for students with sound knowledge bases of both the algorithmic and data-driven approaches. To facilitate this, we propose a framework as the basis for an introductory course to the data-driven approach which will expose students to key concepts while providing a high-level design overview of the processing pipeline. Previous literature has seen proposals for degree programs in data science, including a range of courses to establish the knowledge base students will need to fully grasp this new paradigm [2]. A course built from our proposed framework can serve as the anchor course to such a program, providing students with a broad overview and context to tie in lessons learned in the rest of their coursework. Absent such a specialized degree program, this course would serve as the gateway course in a machine learning or data science concentration within a computer science or engineering degree program.

2 Machine Learning Development Framework

We characterize our framework by three layers of abstraction, shown in Figure 1. Due to the more exploratory nature of the data-driven approach, it is natural to consider the developer as a researcher in this process. Doing so leads the student to shift their mindset more towards a research perspective where they must be asking and answering questions related to the data and the overall problem at multiple levels. This role would also lend itself to prepare students for an interdisciplinary team setting, increasingly common with the globalization of industrial markets [4]. While working in such a group environment, the student developer needs to be cognizant of the variety of engineering factors to be analyzed and communicate with other, potentially non-engineering team members [5]. Coursework and software to facilitate this team setting and in-

teraction have been created with positive results in previous literature ([3, 8]). Effective communication in this setting is a critical skill that can be applied in the industrial setting while working with a client to explore and clarify their needs, which often must be translated from domain specific vernacular to the engineering domain. Thus, our framework for an introductory course in data-driven methodology includes ample opportunity for the students to interact with and gather feedback from others at each layer.

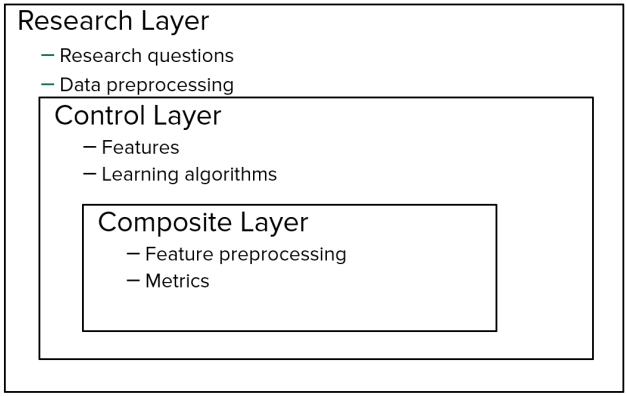


Figure 1: High Level Design

Beginning with a high-level view, our framework guides the researcher through multiple layers of abstraction encompassing the machine learning processing pipeline. The traditional mindset and pipeline as seen in previous literature commits the researcher to a particular solution, locking them into that specific implementation within the vast solution space, often without justification. This unnecessary restriction often leads to sub-optimal results due to current limitations in valid methods of predicting the optimal algorithm [10]. Our framework addresses this problem directly by forcing the researcher to consider alternatives at each layer, guiding them to more thoroughly explore that solution space. At the highest level of abstraction, the research layer forces the researcher to consider the possibility that multiple research questions could be tested using the same data set, perhaps exploring the same research topic but from different perspectives. Preprocessing of the data in different ways could also be considered in its potential to influence the results of the research. Descending into the control layer, the researcher would then be directed toward possible feature sets and subsets thereof before generating a set of possible machine learning algorithms for testing. Students would be instructed as to how this layer would also provide guidance in how to paral-

lelize the process of building and testing the resulting models, leading into the composite layer. Here, the researcher would be faced with the variety of ways the features can be preprocessed before model construction. Different metrics in testing methodology should also be considered as related to the research questions decided upon at the top level.

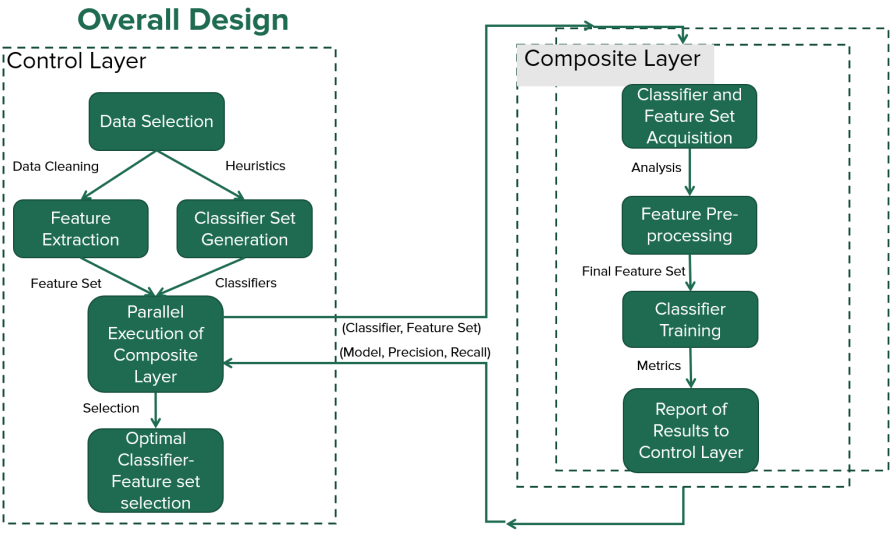


Figure 2: Control and Composite Layer Internal Structure

Establishing a three-layer architecture can naturally be thought of in a hierarchical way: The lower, control layer involves the aforementioned testing methodology, which is constructed as a set of classification problems. For every classification problem, the layer includes a user-defined feature set and set of algorithms. Every algorithm is then trained and tested with a variety of feature subsets. One pairing of an algorithm with feature subset is selected based on performance metric(s) and deemed the empirically best solution to the current classification problem. The control layer repeats this process for all classification problems until it arrives at a set of best solutions for those problems.

This solution set is contained within the control layer, which encapsulates them into a single package that is prepared to accept an unknown instance from the subject domain. The control layer extracts the appropriate features from this instance of interest and provides them to the package, distributing the features to each trained algorithm according to the algorithm-feature pairings. Each algorithm in this ensemble then produces a classification inde-

pendent of each other with the complete set of classifications provided to the user afterward. This overall approach allows for a great degree of flexibility as each algorithm in the ensemble can adapt to the needs of each problem. The problems themselves can also be complementary in the subject domain or completely independent, allowing for further flexibility in adapting to the needs of the domain user.

3 Summary

With the increasing prevalence of data-driven problem solving, computer science curricula stand to benefit from including additional coursework specifically geared for that approach. In this paper, we proposed a conceptual framework to form the basis of a course for data-driven problem solving which would introduce them to critical concepts and provide a high-level design. Through this course, the curricula can show students how to adopt the machine learning paradigm and serve as students' jumping off point into the world of big data, machine learning, and the like.

4 Future Work

Continual development of this framework is ongoing to establish it as the conceptual design of a development environment built from the ground up with data-driven methodology in mind. In addition, we have tested this framework as a machine learning model applied to a case study involving the localization of the seizure onset zone within the brain of epilepsy patients. Early results show great promise when compared to the limited success seen by previous studies [7]. Preliminary results show higher accuracy and the ability of our framework to facilitate the application of data-driven research for parallel research projects. In the interim, our framework can serve as both a conceptual design for our proposed introductory course into machine learning and data-driven problem solving as well as a high-level model for students to follow in future projects to facilitate full adoption of the machine learning paradigm. This embracing of the machine learning paradigm would be further enhanced by using a development environment constructed by implementing our proposed framework. Thus, our framework can serve as the basis for the course to teach students, a model for students to follow, and the underlying architecture for the tool students can use for their machine learning needs.

References

[1] Kent Beck, Mike Hendrickson, and Martin Fowler. *Planning extreme programming*. Addison-Wesley Professional, 2001.

[2] Ismail Bile Hassan, Thanaa Ghanem, David Jacobson, Simon Jin, Katherine Johnson, Dalia Sulieman, and Wei Wei. Data science curriculum design: A case study. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 529–534, 2021.

[3] Lisa J Burnell, John W Priest, and JB Durrett. Teaching distributed multidisciplinary software development. *IEEE software*, 19(5):86–93, 2002.

[4] James J Duderstadt. Engineering for a changing world. In *Holistic engineering education*, pages 17–35. Springer, 2010.

[5] Atila Ertas, Timothy Maxwell, Vicki P Rainey, and Murat M Tanik. Transformation of higher education: The transdisciplinary approach in engineering. *IEEE Transactions on Education*, 46(2):289–295, 2003.

[6] Elif Eyigoz, Sachin Mathur, Mar Santamaria, Guillermo Cecchi, and Melissa Naylor. Linguistic markers predict onset of alzheimer's disease. *EClinicalMedicine*, 28:100583, 2020.

[7] Jiayang Guo, Kun Yang, Hongyi Liu, Chunli Yin, Jing Xiang, Hailong Li, Rongrong Ji, and Yue Gao. A stacked sparse autoencoder-based detector for automatic identification of neuromagnetic high frequency oscillations in epilepsy. *IEEE transactions on medical imaging*, 37(11):2474–2482, 2018.

[8] Letizia Jaccheri and Guttorm Sindre. Software engineering students meet interdisciplinary project work and art. In *2007 11th International Conference Information Visualization (IV'07)*, pages 925–934. IEEE, 2007.

[9] Brian Randell. The 1968/69 nato software engineering reports. *History of software engineering*, 37, 1996.

[10] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Developing a Machine Learning Course for Anomaly Detection*

Shyam P. Prabhakar and Leon Jololian
Electrical and Computer Engineering
University of Alabama at Birmingham
Birmingham, AL 35294
shyamp@uab.edu, leon@uab.edu

Abstract

Machine learning has shown its effectiveness in solving many problems for which traditional algorithmic solutions are not easy to find. For the past decade, we observed the rapid emergence of machine learning courses throughout the curricula. However, the focus of many machine learning courses is predominantly on introducing basic algorithms, such as linear regression, logistic regression, neural networks, and K-nearest neighbors, to name a few. There is limited emphasis on data wrangling, the process by which cleaning and unifying messy and complex data sets for easy access and analysis. In this paper, we discuss the introduction of special topics in a course on machine learning geared towards two major ideas: a) data processing techniques, b) introduction to real world scenario with anomaly detection in datasets using machine learning classifier techniques, and b) the augmentation of the data by introducing new data synthetically created. The objective is to raise awareness of the importance of the pre-processing of the data ensuring the quality of the results obtained in the post-processing stage. The work on this paper was a joint collaboration between the University of Alabama at Birmingham and IBM, where the authors currently work.

1 Introduction

Many data science courses devote much emphasis on teaching machine learning algorithms and to a lesser extent on data engineering and data processing. Yet, processing datasets as a preliminary step is widely considered essential for achieving trustworthy results prior to applying machine-learning algorithms. This paper provides the guidelines for introducing data processing and engineering into the curriculum, highlighting data collection, processing, organizing, and analyzing data for anomaly detection. Understanding the data and its characteristics is an important aspect of the machine learning model development. Introduction to Cross Industry Process for Data Mining (CRISP-DM) benefits the curriculum to understand the data flow and processing. The needed skills for developing an understanding of anomaly detection requires an intermediate level of mathematics in the areas of linear algebra, calculus, probability, and statistics.

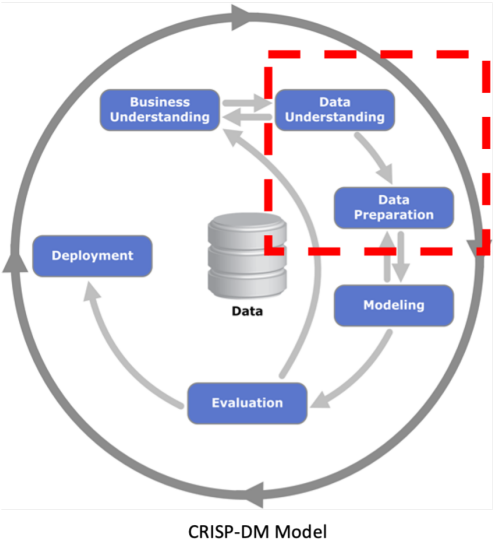


Figure 1: CRISP-DM Model

We define anomalous behavior in a dataset as one that deviate from the known normal. Anomalies are not defined by their own characteristics but in contrast to what is normal. One can build a domain-specific, rule-based system to identify and define normal patterns in the transactional set. The rest of the data in the dataset would be outliers and don't match with the typical behaviors. These sets of data need further analysis to determine if

the distribution possesses normal characteristics or abnormal. Though labeled data is hard to come by, clean, processed, and cataloged data helps in applying machine learning and AI techniques to find the normal and then the plausible anomalous data. Transactional datasets in domains like Finance, Insurance, and Medical have normal and abnormal or anomalous data. Public availability of these datasets is sparse due to various strict federal regulations like HIPAA, BSA AML, etc. Including anomaly detection as a curriculum project helps students apply end-to-end data engineering and data science process.

Main type of anomalies: 1) Point Anomaly: Data instance thar deviates from normal. Most used type in research. One example is credit card fraud detection, 2) Context or Conditional Anomaly: If data instance behaves differently in a context. E.g., If a credit/debit card is used in a different location (state or country), that has no association with the customer. Sudden changes in the spending habits flag anomaly, 3) Collective Anomaly: If a collection of related data instances is anomalous with respect to the entire data set. E.g., If ftp, ssh, and buffer-overflow happen together, it will flag as anomalous in a network intrusion. An individual data instance may not be anomalous.

Challenges in anomaly detection process are the following. 1) The boundary between the normal and anomalous behavior is often not precise, 2) New normal evolve and current normal may not be the same as future normal, 3) Availability of labeled data for training/validation of models are rare. Often the information is noisy and difficult to identify and remove, 4) The abnormal data is often rare and much smaller in volume when compared to the normal data. Therefore, to achieve good results often requires that we augmented the data with new synthetically created anomalous data that is statistically consistent with the provided anomalous patterns in the dataset.

Generative models like Variational Autoencoder (VAE) and Generative Adversarial Network (GAN) demonstrated that they could recreate or regenerate and modify specific input data characteristics, especially with image data. Therefore, our course will include material that explains generative models like general auto encoders, VAE and GAN.

Anomaly detection is done first as it needs to find out what you need to look for in normal behavior. The use of anomaly detection models such as point anomaly detection, context or conditional anomaly detection, and collective anomaly detection methodology helps in identifying outliers efficiently. The next step would be to train and implement an efficient classification model to assign new examples to the categories that you have already identified. Update the anomaly detector with the new examples and repeat the process. Logistic Regression, SVM, Random Forrest, and XGBoost Classifiers are some of the commonly used classification models. Figure 2 shows an anomaly detection process that could be used in the initial stage of discovery process.

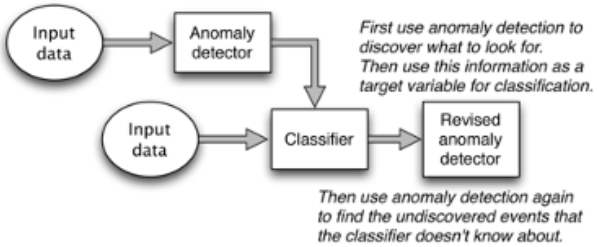


Figure 2: A high level Anomaly Detection Process (Courtesy: [3])

2 Literature References of Anomaly Detection and Data Augmentation

Focus of all the main research in the field of anomaly detection is to use Machine Learning or Deep Learning classifier to classify normal and abnormal data behaviors. Various techniques in machine learning, deep learning, statistical, and rule-based methods are available to detect frauds, information security, and electronic crimes. These techniques address domain-specific anomaly detection and are based on finding outliers in a single instance, context-based, and collective anomaly. Varun Chandola, Arindam Banerjee, and Vipin Kumar (2009) [1] provided an extensive survey on Anomaly detection where they lay out and compare all the anomaly detection techniques developed until 2009.

Many proposed pattern detection methods are optimized to detect abnormal patterns in data from a specific domain (Chau et al., 2006; Neill and Cooper, 2010; Neill, 2011) [2, 10, 9, 8]. Das and Schneider (2007) used Bayesian Network (BN) [4] anomaly detector to detect anomalies at the individual record by computing likelihood of each record to the base and assumes that records with lowest likelihood records are most anomalous. This process loose power to detect anomalous groups produced by a subtle anomalous process where each record, when considered individually, is only slightly anomalous. Also, BN ignores the group structure of anomalies and thus fails to provide specific details useful for understanding the underlying anomalous processes. Das et al., 2008 [2], created another model by improving BN called Anomaly Pattern detection (APD) where APD allows looking for anomalies in a subset of records rather than individual records. Other studies based on Bayesian Networks are Anomaly Group Detection (AGD) by Neill et al., 2008; Das, 2009 [2] and FGSS (Fast Generalized Subset Scan) by Edward McFowland III, Skyler Speakman, and Daniel B. Neill (2013) [7]. These models combine BN with statistical methods to identify anomalous patterns in a group. Since AGD has a large search space for the rules, the search would be more time consuming. APD uses 2

component rule for search and AGD uses greedy heuristic search, and this will lose the ability to identify most interesting subset of the data. Though FGSS can efficiently maximize a scoring function over all possible subsets of data records and attributes, allowing to find most anomalous data. The limitation of FGSS is that it will not work on continuous, unstructured, and high dimensional data such as images. We have provided a summary of the research in the anomaly pattern detection domain. All the above research provides a great base to start identifying the abnormal by assuming the models know what the real normal patterns are.

3 Suggested topics for a Course Syllabus

3.1 Data Engineering and Processing

An AI ladder includes data collecting, organizing, and analyzing process. Collection process is where all the data are collected from various sources systems and external sources. The data can be collected in various databases like IBM DB2, Oracle, MS SQL and big data file systems like HDFS (Hadoop Distributed File Systems) and GPFS (General Parallel File System). Organizing process is done through various Extract Transform and Load (ETL) tools like Apache SPARK, Apache KAFKA, and IBM DataStage. In this process we define and create various data stores to summarize and organize the data for easy access. Analyzing process is where we discover and develop strong feature list for machine learning modeling. Efficient feature list with labels helps the classification models to perform better.

3.2 Anomaly Detection with Supervised Learning: Classic Classification Machine Learning Algorithms

In this section we provide lab exercises to help the student in familiarizing with the data pre-processing, feature engineering and modeling supervised learning classifiers to detect anomalies. As discussed before, public datasets are not available due to regulation in vast majority of the domains, so simulating data or using dataset from known domains like Kaggle will help start the process. We will cover the simulated synthetic data in a little detail in the next section. Here we are using credit card fraud detection data [11], PaySim [6]: A ***synthetic data*** that simulates mobile money transactions and KDD cup Intrusion detection data [5] to show how various machine learning classifiers performed.

Deep Learning models can be also used in detecting anomalies. Like, Autoencoder, would be an ideal neural network to learn a typical or normal data behaviors. Any test data that deviates from the normal would be considered as anomalous. We can use the loss functions like MSE, Cross Entropy, BCE

Credit Card Fraud Data

Model	PR	Recall	F1 Score	ROC /AUC
Random Forrest Classifier	0.93	0.78	0.85	88.90%
AdaBoost Classifier (Adaptive Booster Classifier)	0.85	0.7	0.77	84.98%
XGBoost Classifier	0.82			97.86%

KDD Intrusion Detection Data

Guassian NaiveBayes	dos (smurf, teardrop, back)	0.98	0.94	0.96	Train Score: 87.93%
	normal (normal)	0.97	0.64	0.77	Test Score: 87.91%
	probe(ipsweep, nmap)	0.09	0.99	0.17	Accuracy Score: 87.91%
	r21(ftp_write, guess_passwd, phf)	0.29	0.39	0.33	
	u2r(rootkit, spy)	0.01	0.74	0.01	
Random Forrest	dos (smurf, teardrop, back)	1.00	1.00	1.00	Train Score: 99.99%
	normal (normal)	1.00	1.00	1.00	Test Score: 99.97%
	probe(ipsweep, nmap)	1.00	0.99	0.99	Accuracy Score: 99.97%
	r21(ftp_write, guess_passwd, phf)	0.99	0.96	0.98	
	u2r(rootkit, spy)	1.00	0.68	0.81	
Logistic Regression	dos (smurf, teardrop, back)	1.00	1.00	1.00	Train Score: 99.99%
	normal (normal)	0.98	1.00	0.99	Test Score: 99.97%
	probe(ipsweep, nmap)	0.98	0.91	0.94	Accuracy Score: 99.94%
	r21(ftp_write, guess_passwd, phf)	0.84	0.82	0.83	
	u2r(rootkit, spy)	1.00	0.32	0.48	
Support Vector	dos (smurf, teardrop, back)	1.00	1.00	1.00	Train Score: 99.99%
	normal (normal)	1.00	1.00	1.00	Test Score: 99.97%
	probe(ipsweep, nmap)	0.99	0.97	0.98	Accuracy Score: 99.98%
	r21(ftp_write, guess_passwd, phf)	0.88	0.87	0.88	
	u2r(rootkit, spy)	0.80	0.63	0.71	

PaySIm Data

Model	PR	Recall	F1 Score	ROC /AUC
Random Forrest Classifier (sklearn)	0.98	0.78	0.87	89%
Artificial Neural Network with ScikitLearn MLP Classifier (Multi Layer Perceptron)	0.84	0.65	0.73	82.37%
AdaBoost Classifier (Adaptive Booster Classifier)	0.88	0.35	0.5	67.36%
AdaBoost Classifier (Adaptive Decision tree)	0.88	0.35	0.5	68.84%
XGBoost				92.38%

Figure 3: Classification Model testing outputs

Loss, and L1 Loss to determine the possible outliers. We tested this with the credit card fraud data [11] and the outputs are shown below, in Figure 4. We got a better performance with the neural network Autoencoder and used credit card fraud data to test the hypothesis.

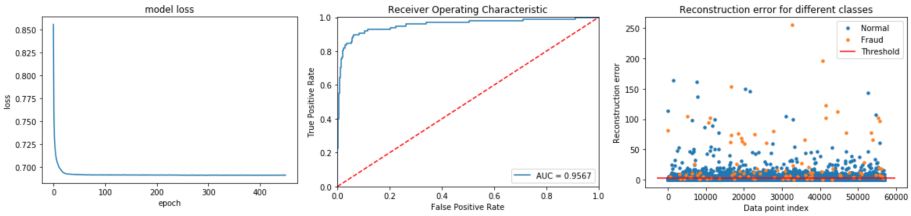


Figure 4: Outputs from the Autoencoder

3.3 Synthetic Data Creation

Synthetic data creation is a process of creating similar data that we can use to train a model when the data is available in a limited quantity or unavailable.

In anomaly detection scenario, not only the source data is limited but also the anomalous events are rare, the data would be very small in numbers when compared to the normal.

It would be point to ponder the ideas of generating this rare events or anomalous data probability distribution and also creating the distribution that has close resemblance with the rare events that could occur in the future. This would help in anomaly detection by predicting futuristic anomalous behaviors. Our research further extends to these aspects and future studies. Introduction of deep generative models like Variational Autoencoders (VAE) and Generative Adversarial Network (GAN) would provide the student with advanced model development abilities in the situation where they have to depend on limited data.

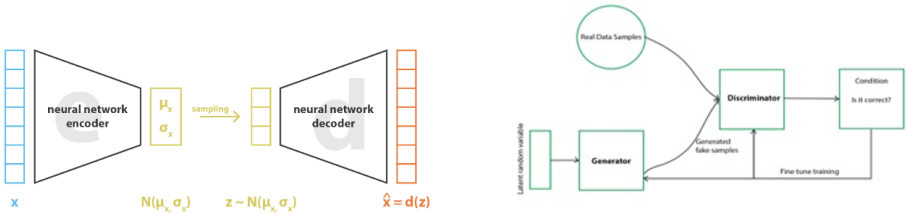


Figure 5: Showing Variational Autoencoders (VAE) and Generative Adversarial Network (GAN)

4 Conclusion and Future Work

In this paper, we suggest the inclusion of topics in data preprocessing in machine learning courses. We discussed anomaly detection and data augmentation as two necessary steps in preprocessing of datasets for machine learning applications. We characterize abnormal data as one that deviates substantially from the majority of the data. The detection and removal of the anomalous data from the dataset leads to a higher degree of accuracy in the machine learning models being developed. A related problem is how to augment the data when the dataset does not include enough samples of the anomalous data. We show how machine learning techniques such as Generative Adversarial Networks and Variational Autoencoders are used to synthetically create anomalous data.

References

[1] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[2] Kaustav Das, Jeff Schneider, and Daniel B Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–176, 2008.

[3] Ted Dunning and Ellen Friedman. *Practical Machine Learning: Innovations in Recommendation*. " O'Reilly Media, Inc.", 2014.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] The UCI KDD Archive Information and Computer Science. Kdd cup 1999 data.

[6] Edgar Lopez-Rojas. Synthetic financial datasets for fraud detection.

[7] Edward McFowland, Skyler Speakman, and Daniel B Neill. Fast generalized subset scan for anomalous pattern detection. *The Journal of Machine Learning Research*, 14(1):1533–1561, 2013.

[8] Daniel B Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):337–360, 2012.

[9] Daniel B Neill and Gregory F Cooper. A multivariate bayesian scan statistic for early event detection and characterization. *Machine learning*, 79(3):261–282, 2010.

[10] Daniel B Neill and Jeff Lingwall. A nonparametric scan statistic for multivariate disease surveillance. *Advances in Disease Surveillance*, 4(106):570, 2007.

[11] Machine Learning Group ULB. Credit card fraud detection.

Wrapper Algorithm for Choosing Machine Learning Functions and Methods in SSAS*

Kelsey Buckles[1], Eduardo Bezerra[2], Eduardo Ogasawara[2], and Mario Guimaraes[3]
[1]NASA & Saint Martin's University, Huntsville, AL
kelsey.d.buckles@nasa.gov
[2]CEFET/RJ, Rio de Janeiro, Brazil
ebezerra@cefet-rj.br, eogasawara@ieee.org
[3]Saint Martin's University, Lacey, WA
MGuimaraes@stmartin.edu

Abstract

Installing and using machine learning software has become significantly more accessible over the years. The abundance of data, the affordability of computer power, and the availability of literature and quality data mining software have made it easier to use data mining tools. The cloud has reduced the difficulty of installing these tools. Meanwhile, the abundance of available data mining functions and algorithms options has created a major challenge for selecting adequate ones. Despite literature about selecting the best machine learning algorithm for a given mining function, there are still opportunities for improvement and we believe it is an essential topic for any Machine Learning/Data Mining class. This paper presents a *wrapper algorithm* that looks at the data and desired outcome (regarding what the user would like to glean from the given data) and chooses an optimal method for data mining. We have evaluated this approach targeting Microsoft SSAS, however the framework can be used for any environment.

1 Problem and Motivation

Consider the research scenario of studying a new dataset to find novel patterns and relationships among attributes. In this case, we are not only targeting to find a proper data mining method. We are interested in discovering which data mining function (prediction, clustering, or pattern mining) to apply.

With many different data mining and machine learning methods available for various applications, one might have difficulty choosing the proper methods to use. Such problems can be decreased for prediction problems in the presence of tools such as Auto Weka [6]. However, when it comes to combining the choice of data mining function with data mining methods, we lack tools. Having some level of automatization to drive users to find appropriate data mining functions and initial data mining methods becomes a productive approach to get insights from data.

Even though Microsoft SSAS has an easy-to-use interface for implementing the data mining methods, some intuition is required on the user's part regarding what data mining function and method to use. This paper presents a *wrapper algorithm* that looks at the data and desired outcome (regarding what the user would like to glean from the given data) and chooses an optimal method for data mining. We have evaluated this approach targeting Microsoft SSAS. We will expand this approach for any machine learning/data mining environment.

Besides the introduction, this paper is organized into three more sections. Section 2 provides generally related literature. Section 3 provides baseline data mining methods available at SASS. Finally, Section 4 presents the wrapper algorithm for data mining function and methods using Microsoft SSAS.

2 Related Literature

Consider the scenario of finding proper data mining methods for a given data mining function. We can find solutions driven to non-expert users such as Auto-WEKA [6]. It considers the problem of finding algorithms and setting their hyperparameters. It applies grid search hyperparameter optimization for each available data mining method. It is computationally expensive and a black box solution, which does not explain the non-expert strategy adopted while selecting the algorithm.

After a cursory search, it appears there is not much research into selecting the proper method for *any* data mining application. It appears there was a project that intended to rank the best data mining algorithms for any application called ESPRIT METAL, but the project itself is no longer available online [2].

Most research available on the topic investigates either the accuracy of

different algorithms on the same data set or compares two algorithms using a handful of applications [13].

3 Baseline data mining methods in SASS

First, to develop a way to assign a data mining method to the data, we must define the different data mining methods available, their strengths and weaknesses, and which mining functions are commonly applied.

Decision Tree. A decision tree algorithm uses information theory to create an accurate tree driven to a classification problem. It categorizes all the attributes into predictable bins. In addition to being easy to understand (white box algorithm), decision trees are helpful when you need to traverse the entire solution space. However, decision trees can become relatively unstable when they are deep [15]. In this case, a small perturbation in the data can change the resulting decision tree. Generally, the use of Random Forests helps find robust optimal trees, but it appears SSAS does not use Random Forests [9].

K-Means. The goal of the K-Means algorithm is to partition the data into groups. The k in K-Means refers to the user-defined number of data points that originally seed the solution space. The algorithm works by first seeding this data and then iteratively refining the corresponding clusters. At the same time, new points are being thrown down into the solution space by minimizing the differences between members of one cluster and maximizing the distance between all the clusters formed. Although the clustering problem is computationally expensive (NP-hard) [5], heuristic methods reduce this cost.

Expectation-Maximization. Expectation maximization is an iterative process that uses optimization to find the best parameters to model the probability distribution of the data [3]. A particularity of this algorithm is that it can build models that depend on latent (unobserved) variables. This algorithm is commonly used for data clustering [8] and computer vision. As a clustering algorithm, EM can be viewed as a generalization to K-Means, because it can fit multi-dimensional Gaussians to the training data. Natural language processing applications are also common. The algorithm can be very slow and works best when you only have a small percentage of missing data, and the dimensionality of the data is not too large [1].

Apriori algorithm. The Apriori algorithm generates and counts candidate itemsets [7] to identify underlying patterns. The Apriori algorithm is traditionally used to find patterns in data sets that are collections of transactions, each transaction being a set of items taken from a fixed set. A bottom-up algorithm identifies frequent individual items in the transactions, creating a subset of these frequent items. Although the algorithm is robust and historically significant, it is inefficient in that it scans through the transactions too

many times, reducing overall performance.

Linear Regression. Linear regression models the relationship between a set of predictor variables and a dependent variable. Microsoft SSAS currently only supports linear regression with two predictor variables to create a linear regression line [10] but could be expanded to three (creating a plane) or more variables with ease. The algorithm fits a line to the training data by minimizing the squared mean distance between the predicted values and the teaching signal stored in the dependent variable. However, it can oversimplify relationships. Not all relationships are linear, and not all predictors are of equal weight, distorting the predictions obtained from the linear regression model. This model has fast performance even on large sets of data. Although it is notorious for oversimplifying predictions, in some fields, one can expect a linear relationship, like a supply and demand model, and in those situations, it performs exceptionally well.

Logistic Regression. Logistic regression is like linear regression, except it uses the sigmoid function to model the relationship between the predictors and the dependent variable, which is categorical [11]. Applications of the vanilla version of logistic regression algorithm must be binary classification problems, although there are extensions to multiclass classifications problems. Additionally, the algorithm will not automatically reduce noise, so some cleaning of the data to remove outliers might be required in the preprocessing phase. Similarly, it can overfit if there are too many data points that are too similar. One might also consider removing these duplicates during preprocessing. The algorithm can fail to converge, especially if the data is sparse or has many highly correlated inputs [4]. Logistic regression is easier to interpret and very efficient to train.

Naive Bayes Algorithm. Naive Bayes algorithm is a classification algorithm that assumes features of examples are conditionally independent given the class. Hence, the simplifying assumption is that none of the characteristics depend on each other, giving the moniker 'Naive' to the algorithm. Naive Bayes works by estimating the prior probability of each class from the training data. It also estimates the likelihoods for each of the predictor attributes. A given example, represented as a feature vector x, can be classified by maximum-a-posteriori estimation at inference time. For each class c, its posterior probability is computed through the following equation: $P(c|x) = P(x|c)P(c)/P(x)$.

In the above expression, $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes), $P(c)$ is the prior probability of a class, $P(x|c)$ is the likelihood of the predictor given a class, and $P(x)$ is the prior probability of the predictor. The predicted class for x will be the one with the greatest posterior probability mass.

The benefit of this algorithm is that it is fast and computationally inex-

pensive. It performs well when there are multiple classes to predict. It also performs well when the input is categorical rather than numerical. If the attributes are independent, Naive Bayes performs well overall and requires less training data than other algorithms [14]. However, if a category is presented in the testing phase that was not shown to the algorithm in the training phase, it cannot make a prediction. Realistically, it is almost impossible for predictors to be completely independent.

Neural Network Algorithm. The neural network algorithm works by calculating the probabilities of each combination of each possible state of the input attribute against each possible state of the predictable attribute [12]. The network can be used for association analysis and can include multiple outputs. The algorithm performs well when analyzing highly complex data if a significant amount of training data is available. Classic applications of neural networks include image processing, natural language processing, and pattern recognition.

Summary. Each of these algorithms is used within Microsoft SSAS under some Microsoft-specific names. Some of the Microsoft names are the names of the actual algorithms, as in the case of Decision Trees, Naive Bayes, Neural Networks, Linear Regression, and Logistic Regression. Microsoft also provides some direction as to which method to use in specific applications. Figure 1 displays an infographic for a quick breakdown of how these Microsoft-specific algorithms are related.

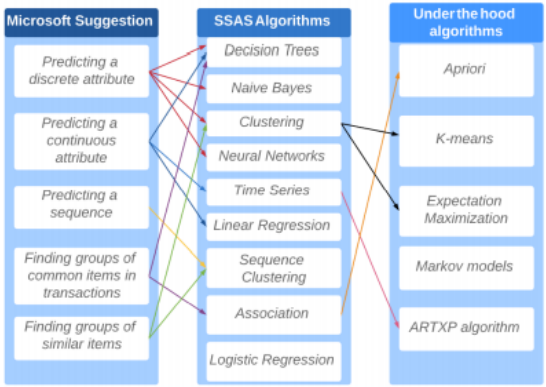


Figure 1: Infographic to explain how the Microsoft SSAS algorithms are used and where they originated. Their SSAS name knows SSAS algorithms with no 'under the hood' algorithm connection

4 Wrapper algorithm

In this section, we described our proposed wrapper algorithm. As it was possible to observe in Figure 1 displays, given a mining function, the infographic lists the available mining methods available at SAS. However, the wrapper algorithm differs as the dataset drives it. Instead of starting from the data mining method, the algorithm indicates possible data mining functions from the dataset structure.

As each algorithm is application-dependent, the approach of this wrapper algorithm is to start with a complete set of every algorithm available in MS SSAS and then pick off algorithms that are not suited for that application until only one algorithm remains.

To illustrate how the algorithm chooses which data mining method to use, Figure 2 presents a Decision-Tree flowchart for an example. In the chart, D represents the Decision tree, K is K-Means, EM is Expectation-Maximization, A is Apriori, Lin, and Log for Linear and Logistic Regression (respectfully), NB for Naive Bayes, NN for Neural Networks.

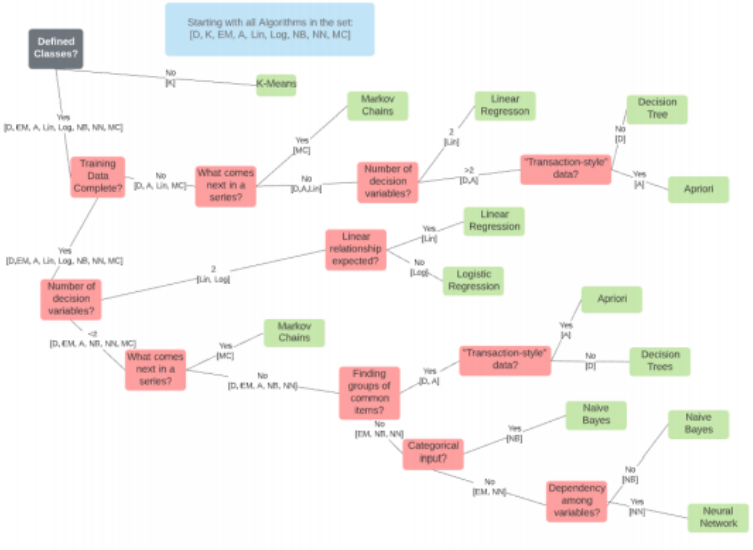


Figure 2: A decision tree is showing how to decide the appropriate algorithm.

The first characteristic of the data mining application to be analyzed should be whether classes are defined. If the classes are not defined, then KMeans would be the only reasonable choice.

If there are defined classes in the data, then the algorithm continues to decide the best data mining algorithm by next looking at the completeness of the training data. Of course, training data sets are not perfect–if they were, it wouldn't be considered machine learning. If the data is primarily incomplete, expectation-maximization, logistic regression, Naive Bayes, and Neural Networks are running.

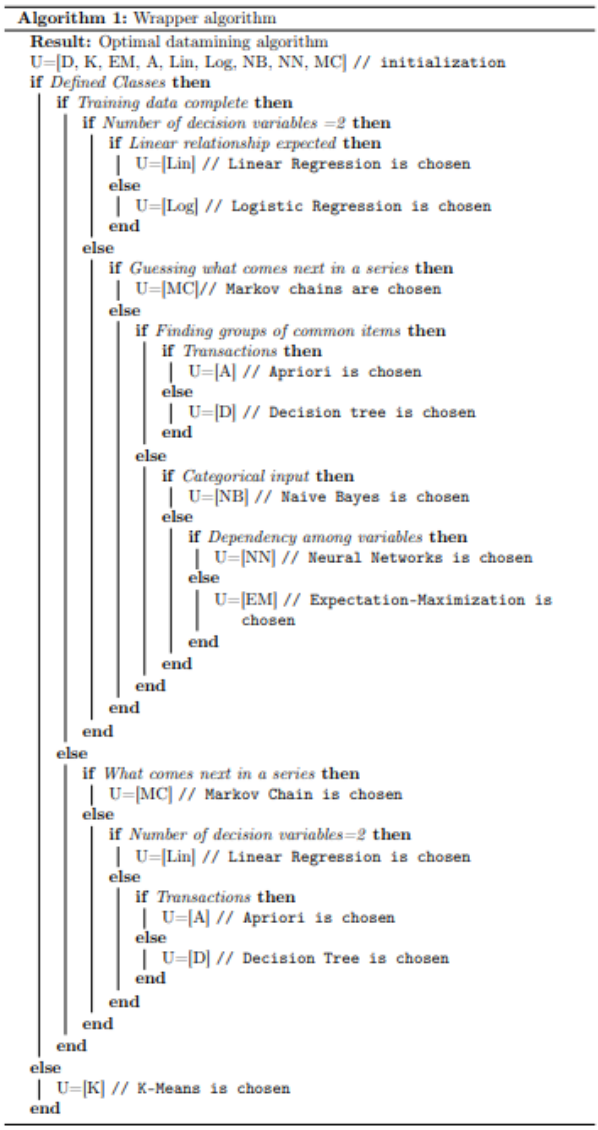
Next would be to look at the number of decision variables. Assuming this algorithm will be distributed with SSAS if the number of decision variables is greater than 2, then Linear and Logistic Regression would be out of the running. If the number of decision variables is 2, then either Linear or Logistic Regression are the best choices. To choose between the 2, if a linear relationship was expected of the data (i.e., the data is studying supply and demand), then linear is the chosen algorithm. If not, then logistics is the chosen one. If the application chooses what comes next in a series, then the chosen algorithm will use Markov chains.

If the application is to find everyday items, then a Decision tree or Apriori is the best choice. To decide between the two, if the data is' transaction' style, then Apriori is the best algorithm. Otherwise, the choice is Decision trees. Lastly, if the input is categorical, then Naive Bayes is the best. Otherwise, Expectation-Maximization or Neural Networks are in the running. According to Nelwamondo, the accuracy of Expectation-Maximization is highly problem-dependent. Still, it seems that Expectation-Maximization performs better when there is minimal dependency among the variables [13].

These ideas can be summed up in the pseudocode described in Figure 3.

5 Conclusion

In this paper, we sketch the design of a wrapper algorithm to choose the appropriate data mining function and methods for a given training dataset. It appears this contribution of a wrapper algorithm for choosing the best MS SSAS algorithm application is unique in the field of data mining. Further research is necessary to implement the algorithm into future releases of MS SSAS. Perhaps a wizard-style format will best suit users' needs in determining the best data mining algorithm to use. Selecting the appropriate algorithm for a specific problem is an essential topic of any machine learning / data mining class and there is a need for more literature and tools.



```
Algorithm 1: Wrapper algorithm
Result: Optimal datamining algorithm
U=[D, K, EM, A, Lin, Log, NB, NN, MC] // initialization
if Defined Classes then
    if Training data complete then
        if Number of decision variables =2 then
            if Linear relationship expected then
                U=[Lin] // Linear Regression is chosen
            else
                U=[Log] // Logistic Regression is chosen
            end
        else
            if Guessing what comes next in a series then
                U=[MC]// Markov chains are chosen
            else
                if Finding groups of common items then
                    if Transactions then
                        U=[A] // Apriori is chosen
                    else
                        U=[D] // Decision tree is chosen
                    end
                else
                    if Categorical input then
                        U=[NB] // Naive Bayes is chosen
                    else
                        if Dependency among variables then
                            U=[NN] // Neural Networks is chosen
                        else
                            U=[EM] // Expectation-Maximization is
                              chosen
                        end
                    end
                end
            end
        end
    else
        if What comes next in a series then
            U=[MC] // Markov Chain is chosen
        else
            if Number of decision variables=2 then
                U=[Lin] // Linear Regression is chosen
            else
                if Transactions then
                    U=[A] // Apriori is chosen
                else
                    U=[D] // Decision Tree is chosen
                end
            end
        end
    end
else
    U=[K] // K-Means is chosen
end
```

Figure 3: Wrapper algorithm that returns mining function and method for a given dataset.

References

[1] Stephanie Andale. Em algorithm (expectation-maximization): Simple definition.

[2] Helmut Berrer, Iain Paterson, and Jörg Keller. Evaluation of machine-learning algorithm ranking advisors. In *In Proceedings of the PKDD-2000 Workshop on DataMining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*. Citeseer, 2000.

[3] Jason Brownlee. A gentle introduction to expectation-maximization (em algorithm).

[4] Jason Brownlee. Logistic regression for machine learning.

[5] MR Garey, David Johnson, and Hans Witsenhausen. The complexity of the generalized lloyd-max problem (corresp.). *IEEE Transactions on Information Theory*, 28(2):255–256, 1982.

[6] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 18(25):1–5, 2017.

[7] Microsoft. Microsoft association algorithm technical reference. Technical report, 2018.

[8] Microsoft. Microsoft clustering algorithm technical reference. Technical report, 2018.

[9] Microsoft. Microsoft decision trees algorithm technical reference. Technical report, 2018.

[10] Microsoft. Microsoft linear regression algorithm technical reference. Technical report, 2018.

[11] Microsoft. Microsoft logistic regression algorithm. Technical report, 2018.

[12] Microsoft. Microsoft neural network algorithm technical reference. Technical report, 2018.

[13] Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, pages 1514–1521, 2007.

[14] Business Analytics Sunil Ray and Intelligence. Learn naive bayes algorithm: Naive bayes classifier examples.

[15] Jay Trivedi. The indecisive decision tree — story of an emotional algorithm (1/2).

Computing-As-Literacy: Cross-Disciplinary Computing for All*

Arianna Meinking, Kanalu Monaco, Zachary Dodds
Harvey Mudd College
Claremont, CA 91711
{ameinking, kmonaco, zdodds}@g.hmc.edu

Abstract

As a means of inquiry and expression, computing has become a literacy across many professional paths. This paper casts a vision for how a small, STEM-focused school supports this role of computing-as-literacy. We share several examples, both future visions and past experiences. We hope to prompt and join discussions that further the reach, use, and enjoyment of computing.

1 Computing beyond CS

More and more, computing is contributing to pursuits beyond software – in fact, beyond CS itself. Computing offers a means of inquiry toward understanding and insight. Thought experiments and live-tunable simulations, for example, offer rich environments in which to build understanding of counterintuitive phenomena (such as special relativity) or surprising interactions (e.g., climate-equilibrium simulations). Beyond inquiry, computing offers an expressive medium for experiences, perhaps tailored to an individual style or a group's priorities. Our shared-media era leaves no doubt: Computing expands humans' aesthetic range. Our goal is that this be accessible to everyone.

2 Computing As Literacy

Computing's ability to advance inquiry and expand expression surpasses older roles as a valuable-specialty and liberal-art: Computing is emerging as a professional literacy. In fact, as a personal literacy, computing is well-established – at times, perhaps, too-well-established!

We posit that computing's role as a professional literacy will deepen in years ahead. This work shares two curriculum-development paths by which our small, STEM-focused program embraces this trend:

- discipline-specific bridges using computing to scaffold student interactions and promote insight in physics, climate science, and mathematics.
- we also share tools for building computing context and community, designed to support an undergraduate-universal computing curriculum.

Both efforts are in process. They reflect our community's support for Computing-As-Literacy among all students and all fields of study, building on deep, cross-disciplinary foundations, e.g., in biology[6] and engineering[4]. The possibilities, it seems, grow faster than we can instantiate them. We look forward to teaming with other institutions on this path!

3 Disciplinary Bridge: Paradoxical Physics

Although physics is an inescapable part of the human experience, it is not intuitive as an intellectual endeavor. First-term students at our institution are "thrown into the deep end" with a half-semester special relativity class. Students solve problems and untangle paradoxes. In one classic problem a spaceship is traveling from the sun to another star[3]. But along the trip, our Sun explodes!

This scenario raises a large family of questions about the *r*elative times at which events take place. When is the explosion perceived? When is the second star reached? Formulas yield "answers," but formulas contribute less to conceptual understanding – understanding our physicists want to nurture – of concepts like time dilation or length contraction[3]. Computing provides a path for students to build those sophisticated, interwoven intuitions.

Those intuitions are visual, dynamic, and geometric. Whether imagining force vectors or watching momentum-conserving collisions, physics education benefits from more than the calculations computing offers. To illustrate this, we have built a simulation of the sun-exploding family of special relativity problems. (This and all of this paper's interactions are available at `myappkanalu.firebaseapp.com`).

When run, the student notices that moving clocks run slower than stationary ones (an example of time dilation) and that the distance between moving

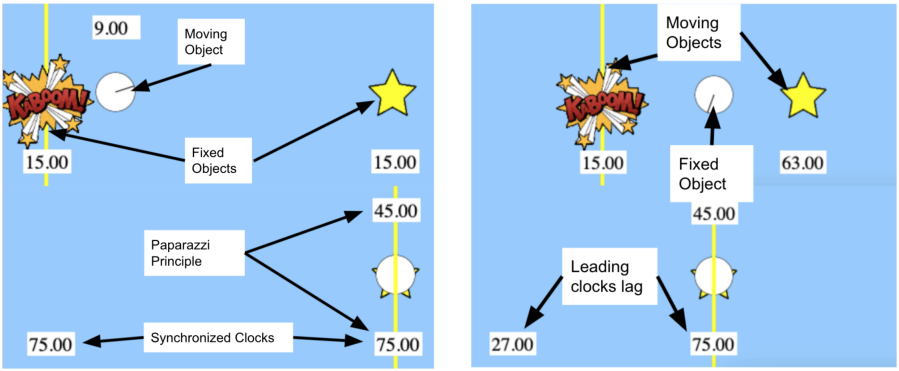


Figure 1: A demonstration of relativity. The left image depicts a simulation in the Sun-and-star's rest frame; both are at rest, and the clock simply travels from the Sun to the star. The right image depicts the same scenario, but in the clock's rest frame; from its point of view, the Sun and star are both moving to the left!

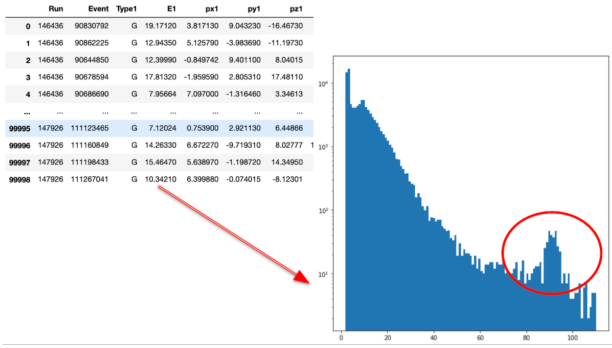
objects is smaller than their distances when not moving (an example of length contraction). Other important concepts, like "leading clocks lag" or the Paparazzi Principle, emerge from this single simulation. And, as physicists like to insist, you'll notice that nothing moves faster than light!

4 Data-Driven Physics

Physical insight and computing share more than simulations. The data-analysis branch of physics depends on computing to create insights – in fact, the computing required is accessible, and adds to student understanding of both CS and physics.

In one such example, students are provided a CERN csv file with 99,999 rows describing distinct particle-collision runs in the Large Hadron Collider[5]. This size is a sweet spot: too much data to process by hand, but a student-with-laptop will succeed! The workflow starts with experiential understanding: making sense of the features across the file's columns using concepts learned in Special Relativity. From there, the data is transformed into a list of masses; these, in turn, are graphed as a histogram. The second half of the challenge incorporates disciplinary insight: students use that histogram to determine the mass of an otherwise-unseen particle created – and destroyed – before it reaches the collision-detector.

Thus, students take data and make it meaningful to them en route to



	Run	Event	Type1	E1	px1	py1	pz1
0	146436	90830792	G	19.17120	3.817130	9.043230	-16.46730
1	146436	90862225	G	12.94350	5.125790	-3.983690	-11.19730
2	146436	90644850	G	12.39990	-0.849742	9.401100	8.04015
3	146436	90678594	G	17.81320	-1.959590	2.805310	17.48110
4	146436	90686690	G	7.95664	7.097000	-1.316460	3.34613
...	...	...	...	...	...	...	...
99995	147926	111123465	G	7.12024	0.753900	2.921130	6.44866
99996	147926	111160849	G	14.26330	6.672270	-9.719310	8.02777
99997	147926	111198433	G	15.46470	5.638970	-1.198720	14.34950
99998	147926	111267041	G	10.34210	6.399880	-0.074015	-8.12301

Figure 2: The physics assignment makes use of CERN's open source data to analyze files. Using equations from Special Relativity and the conservation of energy and momentum, students find masses for each collision in the file. Mass-histogram abnormalities indicate particles that decayed from a "rare" particle: in the assignment, students find this "mystery" particle's mass. Python's matplotlib and pandas are crucial libraries.

analysis and understanding. To cope with a task that has many steps and lots of data, students break down a problem into "helper functions," and mentally, into steps of a repeatable workflow. Such an approach supports not only physics, but real-world scientific and data-handling processes across many fields.

5 Disciplinary Bridge: Mathematics Experienced then Expressed

Like physics, mathematics is a universal requirement at HMC. Many students love math; others disagree; some are in between. Common to all groups is that humans first *do*, then *distill*. Put another way, students only meaningfully express mathematics after they have meaningfully experienced mathematics. Calculations are useful, but computationally-empowered experiences are far more useful for drawing out what mathematicians hope students will share! Here, we show two such example-experiences: the German Tank Problem and the Fenced Random Walk.

6 Teutonic Tanks

The German Tank Problem is a classic statistical thought-experiment. Abstractly, it asks, "What's the maximum?" from a discrete uniform distribution,

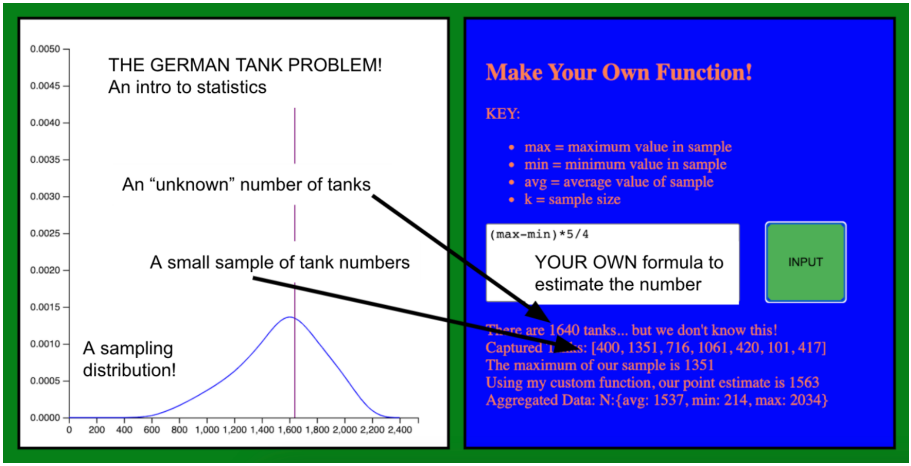


Figure 3: Students can input their own equation to estimate the max. The program will generate over 1000 random samples and calculate an estimate for each one to create a sampling distribution. The one above has decent accuracy and low precision; not so good. The challenge is to create equations that have high accuracy and high precision.

given a small subsample. Its open-endedness and authenticity are its power. During WWII Allied forces wanted to estimate the number of tanks the Axis was producing. Statisticians used the serial numbers from the small number of captured tanks to create estimates – estimates that proved to be much more accurate than traditional intelligence-gathering methods[2].

As an example, suppose seven serial numbers were known/captured: 302, 1953, 1917, 1082, 2176, 1728, and 1401. From these, we want to estimate the maximum – not of those numbers – but of the production-sequence from which they were captured. (In reality, the max was 2329.) Producing an estimate from a single sample-of-seven is not a very effective strategy. Applying another naive solution to this particular sample yields 2486, a bit too high. Using random sampling however, we can create many seven-number-samples less than 2176 (the "observed" max) and track the distribution-max from each. Taking an average of these maxima yields 2328, only one smaller than the real value!

Statistics is an incredibly valuable field, but this problem requires very little knowledge of statistics; in fact there is no "right answer." Our online interaction allows students to create their own formulas and see how well they work! Supported by a statistics curriculum, this reinforces deeper, experiential understandings of accuracy/precision tradeoffs, sampling distributions, and bias.

7 Random (Walk) Insights

Random walks offer students and instructors several points of engagement. In their computing coursework, for instance, all students create a random walker in python (initially an S) that roams back and forth until it hits a wall, returning the number of steps taken. Students build, experiment with, analyze, and extend their simulation. Through many trials, students find that a walker ventures $\sqrt{N}$ steps away from the origin after N random steps. On average, it takes $\sqrt{N}$ steps to reach the wall of the simulation.

This conclusion is powerful, because it is not intuitive: students determine this by relying on experimentation and exercising their computing skills. In CS, students create a fun and simple interface! For instructors, the simulation itself is a door to more. After all, random walks are not limited to the command-line. We have presented the web-versions to high-school and middle-school teachers, who were able to use the interface to deduce the same counterintuitive conclusions.

8 Disciplinary Bridge: Visualizing Interdependencies in Climate Science

"Daisy World" is a climate-science simulation demonstrating how living things affect climate in a hypothetical planet inhabited only by black daisies and white daisies[7]. White daisies reflect a lot of sunlight; black daisies absorb a lot of sunlight (they have different albedos). The Sun grows hotter and hotter over time, but the simulation shows that under certain conditions, the temperature of the planet remains relatively constant for a long period of time.

How is this possible? The presence of daisies has a dramatic impact on the temperature of the planet. Black daisies that absorb heat bloom when the temperature is low, and white daisies that reflect heat replace them as the Sun grows hotter.

Daisy World is a great way to illustrate many fundamental climate concepts such as albedo, feedback systems, and radiative equilibrium. Despite being a relatively simple simulation, there are many functions and calculations needed. The site encapsulates these calculations to emphasize student exploration and understanding of the interdependencies present. Sliders allow students to test how different parameters affect daisy growth and temperature. A map also shows where daisies bloom and provides a different visualization of how species' populations vary over time.

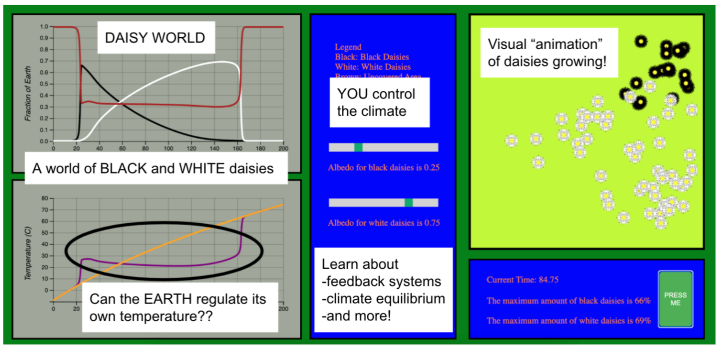


Figure 4: Students have control over simulation parameters and can generate dynamic graphs of daisies vs. time and temperature vs. time. One of the goals of this activity is to explore how to create a temperature equilibrium like the one depicted above.

9 Vision vs. Verdict

We believe these new resources can stand alone as accessible, compelling examples of computational support for science, statistics, and mathematics. As such, they have value as demonstrations, as launchpads to additional analysis, and for building intuition in CS and the bridged field.

Valuable as we hope these examples are, we hope such resources become a larger and larger part of the undergraduate computing experience. As of 2021, building such sites requires more scaffolding than could comfortably fit into a single course. This is also true of the powerful, popular PhET simulations that inspire us[1]. Every year, however, technology and sophistication chip away at these constraints!

In fact, all of these examples here been developed by students – including the coauthors – who researched the available technologies and developed the mindsets needed to leverage them. As computing tools become more accessible and powerful, we promote both the authoring of such simulations and their use as explorations – and insight-generators – in parallel. Authorship and ownership: these are our community's most important scientific resources. As these example resources suggest, computational approaches offer a natural onramp to both.

Acknowledgments The authors gratefully acknowledge the funding support of NSF projects 1707538 and 1612451, along with resources made available by Harvey Mudd College.

References

[1] PhET interactive simulations. `https://phet.colorado.edu/`.

[2] George Clark, Alex Gonye, and Steven J Miller. Lessons from the german tank problem. 2021.

[3] T.M. Helliwell. *Special Relativity.* University Science Books, 2010.

[4] C. F. Van Loan and K. Y. D. Fan. Insight through computing: a matlab introduction to computational science and engineering. 2010.

[5] Thomas McCauley. Dimuon event information derived from the run2010b public mu dataset, 2014.

[6] Parrish Waters and Jennifer A Polack. Interdisciplinary research experience in computer science and biological sciences. *CCSC Eastern*, 36(3):63–69, 2020.

[7] Andrew J. Watson and James E. Lovelock. Biological homeostasis of the global environment: the parable of daisyworld. *Tellus B: Chemical and Physical Meteorology*, 35(4):284–289, 1983.

Curricular and community resources: Supporting Scripting for All*

Lilly Lee, Hallie Seay, Zachary Dodds
Harvey Mudd College
Claremont, CA 91711
{lglee, hseay, dodds}@g.hmc.edu

Abstract

This work envisions resources that help all of an institution's undergraduates build a foundation of computational authorship. Here we present materials evolved from many years of experience requiring Intro-to-Computing (Comp1) of all first-semester students. We hope to prompt and join other institutions looking for ways to engage as much of their undergraduate cohort as possible in computing.

1 Context-and-Community Tools: Developing shared computational models

Every direction we look, our era offers opportunities for computing to contribute. Whether through intensive calculations or insight-producing summaries, computing offers accessible, repeatable, executable interaction-models. From their patterns and dynamics, deeper insights can emerge and fundamental relationships can be discovered or reinforced.

The creation of conceptual models is at the heart of computing. Exploring such models is the realm of introductory computing; our institution requires a "Comp1" course of all first-semester students. This universality has prompted us to develop and customize curricular tools that emphasize students' "shared computing-experience" and promote student-support of each year's new, incoming cohort. This work highlights new directions along this path.

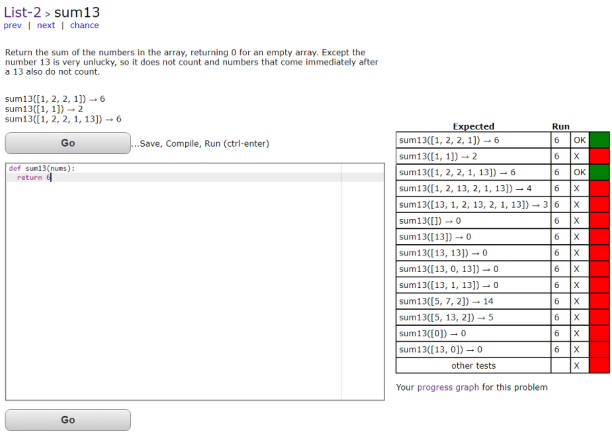


Figure 1: CodingBat interface. It is a clean, simple website with a code editor and test cases with which you can check your code.

2 Coding with Wally: A "multiple-path" CodingBat

For many years our students have used CodingBat, a venerable – and wonderful – set of small-function exercises by Nick Parlante and co-authors at Stanford University[3]. Students are prompted to compose a function matching desired input-output behavior. Once run, students see the test cases they have passed with others, perhaps, failed. Sometimes some of the test cases are hidden.

CodingBat's approach is enormously valuable! It's with good reason that it has become so widespread: to solve a problem, one internalizes the specification, perhaps tries a few examples, understands it, and expresses a solution. Codingbat's interface supports this workflow well.

3 The opportunity: Additional "approach headings" to computational problem-solving

Each of that workflow's components, however, offers opportunity for elaboration. For example, the first step, "understanding the problem," is no trifle! In many cases, understanding the problem *is* the problem – what's more, it is a skill we can actively reinforce.

To that end, we have developed an institution-specific variation on CodingBat named Coding with Wally (Wally is our informal mascot)[1]. Figure 2's example displays Coding with Wally's interface; it is similar to CodingBat in that students are given the prompt, a code editor, and test cases. However, it

Figure 2: Coding with Wally augments CodingBat's approach, offering alternative ways of exploring a problem, its specification, and its input/output behavior. The interface supports the traditional approach of simply writing the function body. The upper-right menu offers several other options: predicting outputs, anticipating inputs, and finding bugs in provided, incorrect versions of the function body itself.

Figure 3: The test input section. Users are given test inputs and are asked to determine the corresponding output. The answer is checked, and the result shared.

differs in *a*lso offering each facet separately.

To focus on the challenge of internalizing what the problem "wants you to do," Coding with Wally offers several alternative interfaces.

Figures 3 and 4 show Coding with Wally's "test input" and "test output" pages of the previously-shown dropdown. Because it is more focused than the "wide-open" writing of a function body like in CodingBat, this mechanic is a powerful one for developing a deeper understanding of the problem. Before diving in, students have the opportunity to step back and carefully consider what the transformation should do, both forward and "in reverse."

This is especially useful for motivating edge cases and/or other details that may escape attention the first time the problem is encountered. It also reinforces that problem understanding is worthwhile – and takes time. Too often anxiety results from the inadvertently-absorbed belief that problem-understanding needs to be immediate: it shouldn't be!

A mechanic that further bolsters understanding is the "test bugs" section. A user is provided a buggy or incomplete version of the code, as shown in Fig. 5.

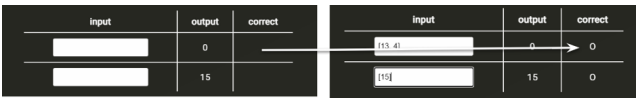


Figure 4: The test output section. Users are given test outputs and are asked to find a possible input (as there can be multiple). Again, the answer is checked and result shared.

Figure 5: The test bugs interface. Users are provided a buggy version of the code at right. The bottom left provides an interface for users to write inputs that reveal the bugs.

By parsing and digesting that code, students are asked to consider how it does – and doesn't – meet the problem specification. This interaction requires deep understanding with the problem specification and, more generally, *p*roblem-specification space! In most cases, comprehending code that someone else has written is more difficult than one's own code. This is all the more true when errors are lurking among its lines. (In this case, the buggy sum13 code does not appropriately consider the numbers that come *a*fter 13!)

Philosophically, the contributions of Coding with Wally echo the insights of Gamage's "Bottom-up" approach[2], in which entire, working artefacts provide a rich, authentic context. From that context, memorable and deeply engaging interactions arise. By approaching a computational challenge in multiple ways, each factored into a purpose-tuned interface, we conspire for each student to develop a nuanced and useful "computational grounding" from which to wrestle with future problems.

4 Slicer: Engaging a particular Python strength

At times, introductory students feel flooded with "sublanguages," those specialized pieces of every language whose expression is worth developing, first

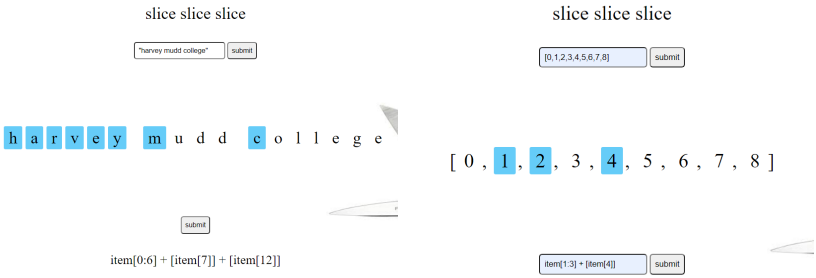


Figure 6: (**left**) Upon inputting a string or array into the first input box, the user is presented with that same input right below in large letters. The user can then "highlight" what they wish to keep by simply clicking on each index. Hitting the second submit button generates a possible combination of concatenated Python slicing to retrieve the indexes highlighted in blue. (**right**) Here, the user inputs their own slicing syntax. The website then highlights what part of the string or array that the user has selected with their code.

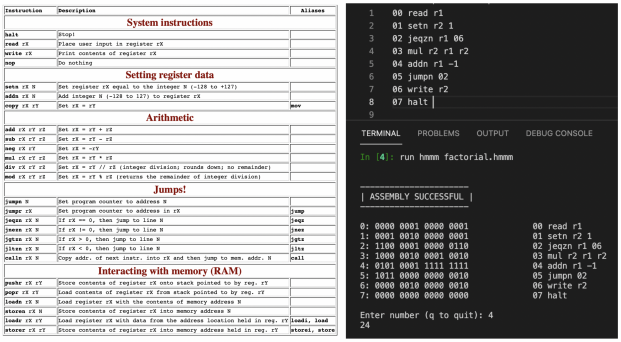
by practicing in isolation, then by integrating into a Coding-with-Wally-type challenge. (From there, it's on to problems "in the wild.") With Slicer, we scaffold this confidence-expanding process.

Python slicing and indexing can be difficult to grasp at first; it constitutes its own mini-language, complete with syntactic and semantic idiosyncrasies worthy of any full-fledged programming idiom! In general, slicing a sequence item takes the format of `item[start:end:stride]`, where `start` represents the starting index, `end` represents the ending index, and `stride` representing the number of indices traversed each step. For example, if `item="abcdef"`, then `item[1:4:2]` would evaluate to `"bd"`. There are many edge cases.

This syntax takes time for newcomers to digest. To help, we introduce *Slicer*, an interactive visual aid that singles out slicing syntax and invites students to build their own conceptual model mapping from that syntax to statement behavior[1]. Figures 6 and 7 show two "directions" for these interactions.

5 Hmmm with Wally: Making "the Machine" Accessible

High-level programming languages frame most of students' "Comp1" interactions. Base-two representation is also part of the experience. Our curriculum further includes a unit on assembly language. We feel assembly valuable-as-knowledge (all software "runs in assembly," after all). It is also further op-



Instruction	Description	Aliases
System instructions		
halt	Stop!	
read *rX*	Place user input in register *rX*	
write *rX*	Print contents of register *rX*	
nop	Do nothing	
Setting register data		
setn *rX N*	Set register *rX* equal to the integer *N* (-128 to +127)	
addn *rX N*	Add integer *N* (-128 to 127) to register *rX*	
copy *rX rY*	Set *rX* = *rY*	**mov**
Arithmetic		
add *rX rY rZ*	Set *rX* = *rY* + *rZ*	
sub *rX rY rZ*	Set *rX* = *rY* − *rZ*	
neg *rX rY*	Set *rX* = −*rY*	
mul *rX rY rZ*	Set *rX* = *rY* * *rZ*	
div *rX rY rZ*	Set *rX* = *rY* // *rZ* (integer division; rounds down; no remainder)	
mod *rX rY rZ*	Set *rX* = *rY* % *rZ* (returns the remainder of integer division)	
Jumps!		
jumpn *N*	Set program counter to address *N*	
jumpr *rX*	Set program counter to address in *rX*	**jump**
jeqzn *rX N*	If *rX* == 0, then jump to line *N*	**jeqz**
jnezn *rX N*	If *rX* != 0, then jump to line *N*	**jnez**
jgtzn *rX N*	If *rX* > 0, then jump to line *N*	**jgtz**
jltzn *rX N*	If *rX* < 0, then jump to line *N*	**jltz**
calln *rX N*	Copy addr. of next instr. into *rX* and then jump to mem. addr. *N*	**call**
Interacting with memory (RAM)		
pushr *rX rY*	Store contents of register *rX* onto stack pointed to by reg. *rY*	
popr *rX rY*	Load contents of register *rX* from stack pointed to by reg. *rY*	
loadn *rX N*	Load register *rX* with the contents of memory address *N*	
storen *rX N*	Store contents of register *rX* into memory address *N*	
loadr *rX rY*	Load register *rX* with data from the address location held in reg. *rY*	**loadi, load**
storer *rX rY*	Store contents of register *rX* into memory address held in reg. *rY*	**storei, store**

```
1  00 read r1
2  01 setn r2 1
3  02 jeqzn r1 06
4  03 mul r2 r1 r2
5  04 addn r1 -1
6  05 jumpn 02
7  06 write r2
8  07 halt
9
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
In [4]: run hmmm factorial.hmmm

| ASSEMBLY SUCCESSFUL |

0: 0000 0001 0000 0001          00 read r1
1: 0001 0010 0000 0001          01 setn r2 1
2: 1100 0001 0000 0110          02 jeqzn r1 06
3: 1000 0010 0001 0010          03 mul r2 r1 r2
4: 0101 0001 1111 1111          04 addn r1 -1
5: 1011 0000 0000 0010          05 jumpn 02
6: 0000 0010 0000 0010          06 write r2
7: 0000 0000 0000 0000          07 halt

Enter number (q to quit): 4
24
```

Figure 7: On the left is a description of each operation in the Hmmm assembly language. On the right is a sample program that takes an input integer and outputs its factorial. This also shows how students traditionally run Hmmm programs through the terminal.

portunity to practice "computational patterns,"[4], i.e., conceptual models of computing processes. There is a cohort-building facet, too, borrowing the spirit of experiments such as [5].

A short, hands-on tour of assembly uncovers a layer of abstraction that enriches the experience of high-level program development, and opens doors to fuller models when the need arises. In a way, assembly is computing's "genetic translation and transcription." Like genetics, it's worth having as part of a computational worldview – even if students don't see their future selves wrestling with machine architectures (or biological ones!)

6 Hmmm...

Thus, every student programs in the assembly language, Hmmm, in their required computing course. Hmmm, the Harvey Mudd miniature machine, is a small, conceptually central subset of all in vivo assemblies. Hmmm digestibly conveys instruction syntax and direct interaction with registers and memory. The machine itself is a 16-register system with 256-memory locations, emulated by a single Python file. Students run, debug, and reason about the Hmmm code they write.

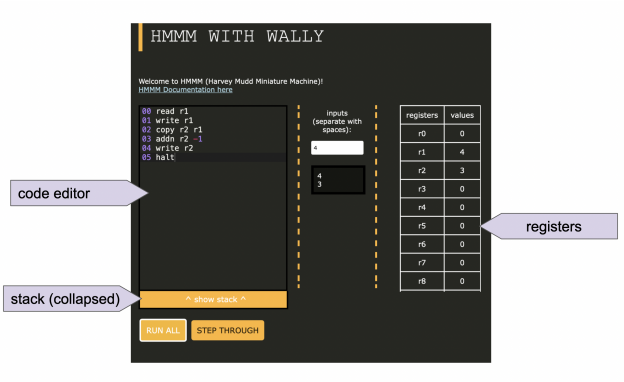


Figure 8: App interface: The page has a simple layout, displaying the editor at left, registers at right, and input/output boxes in the center. The short program shown in the editor was run with the "run all" button. This is similar to how students would previously run their code, with the added benefit of seeing the state of the registers and stack at the end of the execution.

7 Accessible Assembly: Hmmm with Wally

Though Hmmm makes the low-level mindset accessible, all assembly language can be chin-scratching stuff! Beyond the command-line interface, we present here an accessible web application with which students can visualize and tinker with what their Hmmm code is actually doing – and how. This interface reinforces the conceptual model we hope all students take away from their Hmmm experience. Figures 8, 9, and 10 show this interface and summarize its opportunities.

8 Perspective(s)

Sandboxes – where students can focus on one facet of a computational model – offer benefits especially when computing is a universal, shared experience. This work has illustrated the vision – and advantages – of embracing many exploratory and explanatory paths of computing-as-literacy. When building confidence and comfort with a new mindspace, multiple approaches – unpacking problems from different perspectives – offer "onramps" into engagement and understanding. For skills as broadly applicable as computing, this is all the more important. As computing embraces more roles, such approaches are vital: they open doors in all directions, both inward to further computational work and onward across disciplinary specialties.

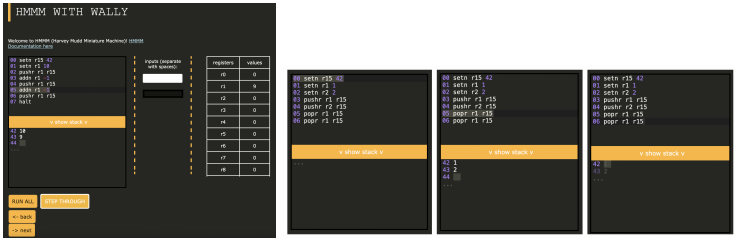


Figure 9: (**left**) "Step through" functionality allows students to run one instruction at a time, moving forward and back incrementally through their code, seeing how registers and the stack changes with each line. Here, the last three lines of code have not yet been run, as reflected in the stack and registers. (**right**) The stack is positioned and changed as an extension of the editor to emphasize how both instructions and the data-stack are stored in the same set of memory locations.

Acknowledgments The authors gratefully acknowledge the funding support of NSF projects 1707538 and 1612451, along with resources made available by Harvey Mudd College.

References

[1] All of the authors' applications in this work are available at https://myappkanalu.firebaseapp.com/.

[2] Lasanthi N. Gamage. A bottom-up approach for computer programming education. *Journal of Computing Sciences in Colleges*, 36:66–75, April 2021.

[3] Nick Parlante. Nifty reflections. *SIGCSE Bulletin*, 39:25–26, 2007.

[4] Richard Rasala and Viera K. Proulx. Pattern and toolkits in introductory cs courses. *Proceedings of the second annual CCSC on Computing in Small Colleges Northwestern conference*, October 2000.

[5] Rita Sperry. We're all in this together: learning communities for first-year computer science majors. *The Journal of Computing Sciences in Colleges and Proceedings of the 32nd Annual CCSC South Central Conference*, page 11–19, April 2021.

Security In Intelligent Home*

Mario Garcia and Yeshihareg Hailu
Southeast Missouri State University
Cape Girardeau, MO 63701
{mgarcia,yfhailu1s}@semo.edu

Abstract

As human needs of intelligent aid are growing higher and higher, people's lives are becoming more dependent on various technologies. One of the fastest-growing Internet of Things' (IoT') technologies, Intelligent Home is becoming more involved in people's lives. Controlling and monitoring home appliances remotely with just a single click or touch of a smart device or a laptop is becoming a common practice. This is possible through IoT like Intelligent/Smart Home System. Since home appliances are required to be connected to internet in order to make them accessible and being monitored remotely, it is obvious that they are vulnerable to cyber-attack. Thus, security and privacy are the main concerns when implementing smart home systems. This paper has discussed and analyzed about security constraints, identify major potential security risks, security constraints, security requirements, the nature of attacks, the security threats at each layer of IoT architecture on smart home and finally design secure smart system.

1 Introduction

The Internet of Things (IoT) refers to the network of devices that are embedded with sensor, software, and other technologies for the purpose of connecting and exchanging data with other devices and system over the internet connection. Therefore, smart homes are benefiting from these IoT technology. It is easy to handle the home lights, switches, doors, cameras, and other electric appliances using a laptop or smart phones with just a single touch.

People can control home appliances from office or anywhere else using their laptop or smart phone. Sensors are used in almost all the connected house appliances. This paper focuses on security issues of intelligent home system and proposed a solution that secure an intelligent home system. Layer based security solution which includes physical protection, implementing smart home firewalls, hub, encryption, and cryptographic strategies are proposed.

2 Problem Description

2.1 Security Requirements

Security and privacy are the main concerns during the implementation of IoT as devices in the IoT system need to be connected to internet in order to be accessed remotely.

The major security requirements of IoT are Confidentiality, Authenticity, Integrity and Availability [11, 19]. Violation of any of these requirements can cause a disaster in the system. Figure 1 illustrates the data security requirements of IoT.

Figure 1: Security Requirements ([1], Figure 2)

2.2 Security Challenges

Fulfilling the above-mentioned security requirements during the implementation of intelligent interconnected systems such as smart home always faces challenges. The major challenges could be Limited devices capabilities[11], Data Management[19], Radio Frequency Identification (RFID)Tags Vulnerability[19], Diverse Communication Protocols[18], Cascading Effect[3, 19], Autonomic control[11, 18, 19] and Physical liability.

Different layers of IoT network architecture are vulnerable for different types of security attacks. Thus, Security attacks can be broadly classified

as Attack based on IoT architecture and RFID and WSN Classifications [11, 18, 19, 5, 6, 10, 17]. Figure 2 illustrates general and layer-based security issues and attacks of IoT architecture.

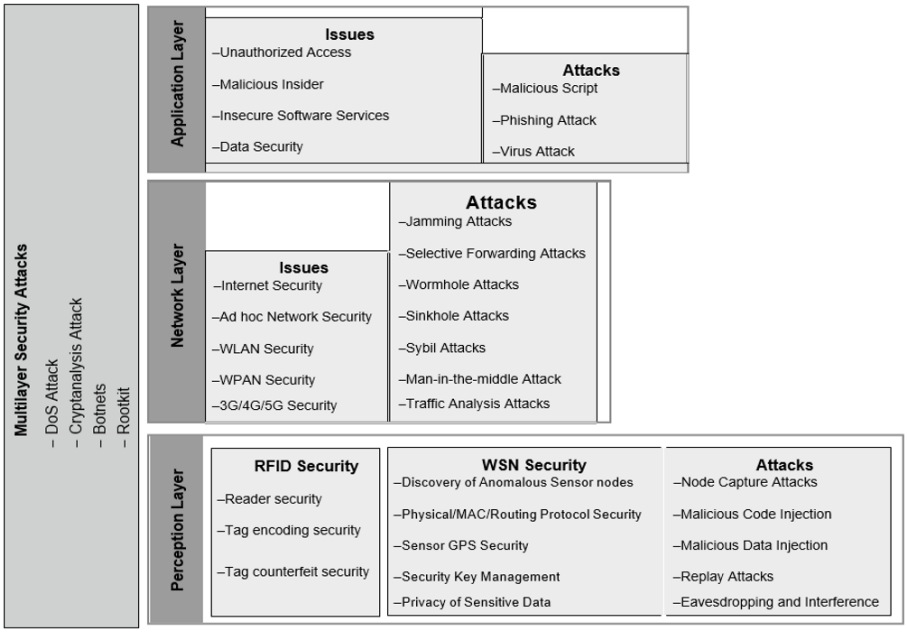


Figure 2: IoT architecture and security issues/attack ([16], Figure 8.3)

2.3 Security Threats and Constraints

When planning to implement IoT system such as smart home, security is always a major concern. The capability of accessing house appliances through the smart home system exposes for cyber-attack and unauthorized access of personal information and data by third party who has special interest about on extracting one's personal data. The IoT system security attack can be classified into main categories as Physical Attack, network attack, Software Attack and Encryption Attack. IP camera hacking through buffer overflow attacks [8], a Distributed Denial-of-Service (DDoS) attack [14], Botnet attack to hack IoT devices [4], SQL injection attacks and cross-site scripting attack [20] are the major potential security threats that IoT is exposed to.

Resource constraints, along with the above mentioned and other types of IoT threats, are the other security problems that IoT system suffering from [15].

Due to lower hardware size of device of IoT, thin operating system installed in some types of IoT devices, and the characteristics of IoT devices which are (heterogeneity, scalability, presence of multiple communication protocols and portability) are barriers of implementing expensive security algorithm, robust communication protocol, dynamic security patches, and conventional security protocol respectively in IoT system.

As mentioned above, Smart Home System is possible through connection of several digital appliances with the use of IoT. Mostly, end users communicate through smart phones to control home appliances. Attacker can easily compromise the users' privacy and security of home devices and related data [7]. Table 1 illustrates security issues of smart home in terms of the vulnerabilities associated with few smart home devices.

Smart device	**Functionality**	**Potential security threat**
Smart Lock	• Lock/unlock without physical key •Lock/unlock through mobile device or web interface • Automatically lock after a specified period of time • Alarms ringing on forced entry or break-ins	• Lock/unlock by attackers to enter/exit from home • Changing of lock/unlock password remotely • Turn off the alarm in case of break-in
Smart Bulb	•Light bulb controllable remotely through mobile application • Scheduling of turning on/off and coloring of light bulbs	• Control the turning on and off behavior of lights • Overload power system by turning on unnecessary lights
Voice Automated Device	• Turn devices on or off based on voice Commands	• Steal private credentials from voice data • Issue voice commands to order unwanted stocks by voice commands • Steal voice data as credentials for use in other voice command systems
Smart Vacuum Cleaner	• Automatically map home layout and conduct automatic and scheduled cleaning in dry or wet mopping modes	• Monitor room activities and stealing of home layout
Smart Refrigerator	• Create grocery list and send order to shops through the Internet • Set expiration data and send related alerts to residents • Suggest recipes based on available ingredients	• Send order with modified grocery list • Modify expiration date of food items in refrigerator or ruin food items by changing temperature
Smart Toilet	• Allow users to remotely set water temperature and pressure • Sense and adjust right water amount to clean itself or for flushing wastes • Notify residents about needed supplies (e.g. toilet paper, soap, and air freshener)	• Turn water tap on and leave water flowing without any need • Remotely control smart toilet's lid and flush nozzles

Table 1: Smart home devices, functionality, and associated security threats ([16], Figure 8.1)

Since smart home system is controlled and monitored via smart mobile

phone Apps like Geolocation Services that can track one's location, the privacy and security of the smart home users are compromised [26]. One of the risks of using this technology is that users are not aware that their location is being tracked by third party. Geolocation and mapping are Apps commonly used by criminals [30].

The other risk is using Wi-Fi. All Wi-Fi clients that were tested were vulnerable to the attack against the group key handshake [28]. Figure 3 shows Smart Home objects internet connection via Wi-Fi.

Figure 3: Smart Home internet connection via Wi-Fi [9]

These limitations, the risks, threats, and constraints of IoT system discussed above, makes implementation of secured IoT system a challenge. Thus, having Efficient Cryptography Techniques, Interoperability, Scalable Solution, Privacy Protection, Resilience to Physical Attacks, Autonomous Control, and Cloud Security are challenges that need to be considered during designing security mechanisms to prevent the risks and threats of personal information misuse.

3 Proposed Solution

For the above-mentioned security attacks, the respective counter measures are described in brief below.

3.1 Physical Attack

To protect the IoT from Physical Attack, use of the correct safety efforts on IoT devices is essential. This is best done by utilizing equipment-based security. Hardware security can also be used to authenticate device ID. This means that a series of security measures can be put between the server and the device itself to establish the authenticity of that device. human-based physical attacks and natural disaster threats are to be addressed and managed as follows:

- Secure sensor design
- Secure sensor deployment
- Secure infrastructure
- Efficient user authentication approach (biometric or smart card) to implement for legitimate access to physical devices and confidential information.

- Implement efficient accessibility control mechanisms
- Efficient implementation of trust management
- Efficient hardware failure recovery schemes

3.2 Network Attack

The first and the main measure to protect from network attack is to make sure that only required ports are exposed and available. After that prepare the services that must not be vulnerable to buffer overflow and fuzzing attacks. The other proposed solution to close the security hole of smart home system is using firewall both at software and hardware level. A firewall is a network security device or software that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules. Figure 4 illustrates the smart home secured architecture using firewall.

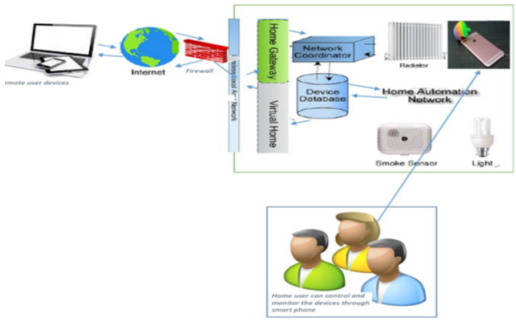


Figure 4: Secure architecture for Smart Home ([27], Figure 2)

The other important proposed solution is using smart home hub [29] network devices. The proposed smart home hub hardware device can connect the appliances which are nodes of the smart home network and provides an intrusion detection system that controls and monitors where and how the information should be exchanged. The device may also include the computing resources that sends a warning message when unauthorized access is detected.

After this, the detail security measure that should be taken to protect network attack is listed below.

- Renaming the router instead of keeping the name provided by the manufacturer.
- Use strong encryption method for Wi-Fi router setting.
- Alter the default username and passwords.

- Setting up a guest network in Wi-Fi access to prevent the private data in the network from being accessed by intruders.
- Disabling the features which are already enabled by the manufacturer like, a remote access if it is not required.
- Change the default privacy and security settings of the devices which are provided by the manufacturer.
- Software updating regularly.
- Avoid connecting to public Wi-Fi networks.
- Two step verifications to easily understand whether the communication is with the genuine sender.
- Use strong, unique passwords for Wi-Fi networks and device accounts: Strong and unique passwords are essential for Wi-Fi networks and devices.
- Disable Telnet login and use SSH [25] where possible.
- Auditing of the IoT devices at regular intervals.
- Have an individual user account for each employee.
- Limited accessibility: Granting limited authority to employees.
- Passwords should be changed at regular intervals.
- Trained cyber security professionals are needed.
- Use proper certification for accessing the internet.

In addition to above mentioned, important technique to prevent the IoT from network attack is connecting the IoT devices to the 0G network because this is a dedicated, low power wireless network specially designed for sending small and critical messages from any IoT device to the Internet [22]. The 0G network does not support network-initiated downlinks, it only supports device-initiated downlinks. These properties make the 0G network not susceptible to network attackers.

3.3 Software Attacks

This attack, which affects the application layer at the top of the three-layer of IoT architecture can be overcome by controls the legitimacy of authorized users (through authentication and access control systems), protection of application software, OS, and end-user interfaces through the utilization of high-level programming languages which assists to avoid insecure programming. Regular updating and installation of antivirus and antispyware software, installing updates that are required by the operating system and application software, taking a backup of business data and information, Controlling Physical access to the system and network components.

3.4 Encryption Attack

The public key infrastructure (PKI) has ensured that the encryption of data must be done through asymmetric and symmetric encryption processes.

3.5 Cryptographic Strategies

Cryptographic algorithms such as symmetric key crypto- graphic algorithms, and advanced encryption standard (AES) [13], Secure hash calculations (SHA) [24], Diffie Hellman (DH) [21], Revest Shamir Adelman (RSA) [12], Elliptic curve cryptography (ECC), and Key Administration are utilized to safeguard information secrecy. Even though the mentioned cryptography algorithms are secure and efficient but that they require more CPU power and consume more battery power. For this reason, they are not a feasible way to verify IoT devices, so there has been an emergence of new cryptographic calculations or advances the existing ones for battery operated IoT devices.

3.6 Authentication and Access Control

The IoT concentrates on a machine to machine (M2M) method of correspondence [2]. Table 2 summarizes the security attacks with counter measures on different layers of the IoT network architecture.

4 Conclusion

This paper has introduced about IoT and smart home which is one of the implementations of IoT in the introduction section. The problem description section identified Security Requirements, Challenges, and discussed in detail. In addition to that, different security Risks, threats and IoT device constraints in terms of hardware, software, and communication protocols which are barrier for implementation of standard security guard are investigated. Cyber-attack and the respective security issues at each layer of IoT have been discussed. Finally, in the proposed Solution section, solutions to fulfill the identified security requirements and to protect the smart home system from the mentioned cyberattack such as physical attack, network attack, software attack, encryption attack, and others are proposed and discussed.

User side layer or application layer	Code injection, data access and authentication	IDS diglossia [19]
	Virus, worms, malware attacks, phishing attacks, spyware	Anti-virus, firewall, IDS [14], secure application code, educating users to use complex passwords, access control mechanisms, key agreement, log monitoring, file and database monitoring tools, anti-malwares to protect applications against malwares.
Support layer security	DoS, wormhole, black hole, interoperability and portability, business continuity and disaster recovery, cloud audit, virtualization security	IDS designed for IoT [15] Lightweight encryption techniques like CLEFIA [22] and PRESENT [17], need for continuous cloud audits, implementation of cloud security alliance standards, secure virtualization technologies, tenant's separation, storage encryption for user's data confidentiality and integrity
Network layer security	Side-channel attacks: Sybil	Malicious firmware/software detection [24], randomized delay [19], intentionally generated noise [20], balancing Hamming weights [21].
	Battery-draining, sleep deprivation attack, routing attacks, node jamming in WSN	Policy-based mechanisms and intrusion detection systems (IDSs)
	RFID tag	Personal RFID firewall [22], anonymous tag, lightweight cryptographic protocol [23]
Physical layer or edge layer security	Node tampering, fake node, malicious code injection, side channel attack,	Authentication with encryption techniques. Physical security in nodes vicinity, need for lightweight encryption algorithms for constrained nodes,
	Mass node authentication, protecting sensor data	authentication and access control mechanisms for devices, anti
	Physical damage	It is better to keep some physical protection for the devices.

Table 2: Summary of attacks on different layers of IoT with counter measures ([23], Table 14.2)

References

[1] Razan AL MOGBIL, Muneerah AL ASQAH, and Salim EL KHEDIRI. Iot: Security challenges and issues of smart homes/cities. In *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, pages 1–6. IEEE, 2020.

[2] Anum Ali, Ghalib A Shah, Muhammad Omer Farooq, and Usman Ghani. Technologies and challenges in developing machine-to-machine applications: A survey. *Journal of Network and Computer Applications*, 83:124–139, 2017.

[3] Bako Ali and Ali Ismail Awad. Cyber and physical security vulnerability assessment for iot-based smart homes. *sensors*, 18(3):817, 2018.

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.

[5] Trevor Braun, Benjamin CM Fung, Farkhund Iqbal, and Babar Shah. Security and privacy challenges in smart cities. *Sustainable cities and society*, 39:499–507, 2018.

[6] Leela Krishna Bysani and Ashok Kumar Turuk. A survey on selective forwarding attack in wireless sensor networks. In *2011 International Conference on Devices and Communications (ICDeCom)*, pages 1–5. IEEE, 2011.

[7] Ziv Chang. Iot device security-locking out risks and threats to smart homes. *Trend Micro Research*, 30, 2019.

[8] R Chirgwin. Get pwned: Web cctv cams can be hijacked by single http request-server buffer overflow equals remote control, 2016.

[9] Ivan Del Pozo and Denise Cangrejo. Creating smart environments: analysis of improving security on smart homes. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 303–310. IEEE, 2018.

[10] Otmane El Mouaatamid, Mohammed Lahmer, and Mostafa Belkasmi. Internet of things security: Layered classification of attacks and possible countermeasures. *electronic journal of information technology*, (9), 2016.

[11] Panagiotis I Radoglou Grammatikis, Panagiotis G Sarigiannidis, and Ioannis D Moscholios. Securing the internet of things: Challenges, threats and solutions. *Internet of Things*, 5:41–70, 2019.

[12] Shay Gueron, Simon Johnson, and Jesse Walker. Sha-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 354–358. IEEE, 2011.

[13] Simon Heron. Advanced encryption standard (aes). *Network Security*, 2009(12):8–12, 2009.

[14] S. Hilton. Dyn analysis summary of friday october 21 attack. `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack`.

[15] Md Mahmud Hossain, Maziar Fotouhi, and Ragib Hasan. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *2015 ieee world congress on services*, pages 21–28. IEEE, 2015.

[16] Muhammad Azhar Iqbal, Sajjad Hussain, Huanlai Xing, and Muhammad Ali Imran. *Enabling the Internet of Things: Fundamentals, Design and Applications*. John Wiley & Sons, 2020.

[17] Hasan Ali Khattak, Munam Ali Shah, Sangeen Khan, Ihsan Ali, and Muhammad Imran. Perception layer security in internet of things. *Future Generation Computer Systems*, 100:144–164, 2019.

[18] Changmin Lee, Luca Zappaterra, Kwanghee Choi, and Hyeong-Ah Choi. Securing smart home: Technologies, security challenges, and security requirements. In *2014 IEEE Conference on Communications and Network Security*, pages 67–72. IEEE, 2014.

[19] Engin Leloglu. A review of security concerns in internet of things. *Journal of Computer and Communications*, 5(1):121–136, 2016.

[20] Zhen Ling, Kaizheng Liu, Yiling Xu, Chao Gao, Yier Jin, Cliff Zou, Xinwen Fu, and Wei Zhao. Iot security: An end-to-end view and case study. *arXiv preprint arXiv:1805.05853*, 2018.

[21] Kevin S McCurley. A key distribution system equivalent to factoring. *Journal of cryptology*, 1(2):95–105, 1988.

[22] M Meraj and Sumit Kumar. Evolution of mobile wireless technology from 0g to 5g. *International Journal of Computer Science and Information Technologies*, 6(3):2545–2551, 2015.

[23] Rajit Nair, Preeti Sharma, and Dileep Kumar Singh. Security attacks in internet of things. *Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications*, pages 237–261, 2020.

[24] Shireen Nisha and Mohammed Farik. Rsa public key cryptography algorithm–a review. *International journal of scientific & technology research*, 6(7):187–191, 2017.

[25] SSH Communications Security. Ssh protocol – secure remote login and file transfer | ssh.com. `https://www.ssh.com`.

[26] Shigeaki Tanimoto, Rei Kinno, Motoi Iwashita, Tohru Kobayashi, Hiroyuki Sato, and Atsushi Kanai. Risk assessment of home gateway/smart meter in smart grid service. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 1126–1131. IEEE, 2016.

[27] Shafiq ur Rehman and Volker Gruhn. An approach to secure smart homes in cyber-physical systems/internet-of-things. In *2018 Fifth International Conference on Software Defined Systems (SDS)*, pages 126–129. IEEE, 2018.

[28] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in wpa2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328, 2017.

[29] Adriana Wilde, Olivia Ojuroye, and Russel Torah. Prototyping a voice-controlled smart home hub wirelessly integrated with a wearable device. In *2015 9th International Conference on Sensing Technology (ICST)*, pages 71–75. IEEE, 2015.

[30] Wang Xi and Luo Ling. Research on iot privacy security risks. In *2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, pages 259–262. IEEE, 2016.

An Introduction to MPI for Python*

Conference Tutorial

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

Because parallel computing has many potential applications, such as databases and data mining, real time simulation of systems, and advanced graphics, it is becoming more and more important. There are two widely known parallel programming models today: message-passing model and shared memory model. In the message-passing model, each task has its private memories, and different tasks can communicate each other via message exchanges. Message Passing Interface (MPI) is a specification designed by a group of researchers from academia and industry and primarily addressing the message-passing model. It specifies the syntax and semantics of a core of library routines useful for message-passing programs and can be implemented in different programming languages, for example, C, Fortran, Java, and Python.

Python is an interpreted, high-level and general-purpose programming language, emphasizing code readability with its notable use of significant indentation. Other than using as a major programming language in CS1 and CS2 courses, Python has also been applied in many areas such as machine learning, artificial intelligence, data science, and data visualization. The package, *mpi4py*, is an MPI for Python package that provides bindings of MPI standard for Python and allows any Python program to exploit multiple processors. This package is constructed on top of the MPI specifications, providing an object-oriented interface similar to MPI-2 C++ bindings. The package *mpi4py* supports point-to-point (i.e., sends, receives) and collective (e.g., broadcasts, scatters, and gathers) communications of any picklable Python object, as well as optimized communications of Python object exposing the single-segment buffer interface (NumPy arrays, built in bytes/string/array objects).

In this tutorial, Python MPI parallel programming with *mpi4py* is introduced. It firstly will introduce where to acquire and how to install the needed

software like *mpi4py* and *Anaconda* on Windows machines and Linux machines. After that, the tutorial will explain how to edit, compile, and run an MPI program in Python on a Windows machine and a Linux machine, including the basic organization of a Python MPI program. Thirdly, it will cover some MPI routines available in *mpi4py* and used for point-to-point and collective communications. Finally, the materials suitable for further study will be listed and described.

The intended audience of this tutorial is anyone who is a beginner to parallel programming and/or is interested in MPI programming, but has learned basic knowledge of the programming languages, such as Python, Java, C#, Fortran, or C++. The expected learning outcomes include the followings. Firstly, after attending the tutorial, the audience should know what software will be needed for MPI parallel programming in Python, especially where to get a copy and how to install on a windows machine or Linux machine. Then, the audience should know what MPI is and what are point-to-point and collective communications. Other than that, how to edit, compile, and run an MPI programs in Python will be learned. Moreover, the audience should learn how to implement various MPI operations in Python, such as point-to-point communications and collective communications. At the end of the tutorial, the e-version of the lecture notes, code in Python as examples, and other materials for self-study can be provided, if needed.

Using Cocalc to Teach Python*

Conference Tutorial

Harold Nelson
Computer Science Department
Saint Martins University
Lacey, WA 98503
hnelson@stmartin.edu

Description

Cocalc (Collaborative Calculation) is a cloud-based system originally created to facilitate the use of SageMath, a computer algebra system. It also provides a platform for other languages, including python. It enables a literate style of programming through the use of Jupyter notebooks. The collaborative feature makes it an ideal system for teaching remotely.

Tutorial Proposal

This hands-on tutorial session will provide a walkthrough of the steps needed to deliver a course: creating a course, a handout, and an assignment. There will also be a demonstration of the literate programming style and collaboration.

Teaching Numerical Methods to Computer Science Majors using SageMath Interacts*

Conference Tutorial

Razvan A. Mezei
The Hal and Inge Marcus School of Engineering
Saint Martin's University, Lacey, WA 98503
rmezei@stmartin.edu

In this tutorial we will demonstrate the use of SageMath Interacts ([2, 3]) to as a great way to implement various numerical methods in a class consisting of mostly undergraduate Computer Science majors (although students can also be non-CS majors). The use of Interacts (SageMath Interacts) is a great and fun way to introduce Computer Science students various Math concepts such as: variables, functions, solving non-linear equations, approximation, polynomial and spline interpolation, etc. Such Interacts help students visualize the problems they are working on, as well as give them a chance to easily program nice Graphical User Interface applications. SageMath is an open-source Python-based tool, so learners will get exposed to the syntax of Python and will also make use of the symbolic computation capabilities provided by SageMath. In this tutorial we will explore all the above topics through hands-on applications. While the tutorial will focus on programming examples used in a Numerical Methods course ([1]), they can be used in other courses such as: Introduction to Computer Science, Discrete Mathematics, and other.

References

[1] G.A. Anastassiou and R. A. Mezei. *Numerical Analysis Using Sage*. Springer, 2015.

[2] G.V. Bard. Sage for undergraduates (online version).

[3] W.A. Stein et al. Sagemath - open-source mathematical software system.

Teaching Computer Science in 3D Virtual Worlds*

Conference Tutorial

Cynthia Calongne
St. Martin's University
Lacey, WA 98503
calongne@pcisys.net

Virtual worlds are 3D environments where educators simulate real or imagined online learning spaces and teach classes within them. During a pandemic, it was a delightful experience to teach synchronous classes on a virtual beach or within a French village and to watch the student projects come to life.

Considerable work goes into planning these imaginative classrooms. This tutorial describes the experiences from past classes taught in virtual worlds and provides examples from a case study for a virtual reality class hosted in the Fall 2020 term for a Saint Martin's University class. It features the technology, design of the learning environment, and the course design activities as well as educator observations.

The goal is to identify the requirements and process for planning and hosting a computer science class in a 3D virtual world.

For visual examples from past classes and conference presentations, visit the repository hosted online as provided in the references[2] and review the images from the Fall 2020 class[3].

Computer science topics taught in virtual worlds featured the following.

- software requirements engineering
- software design
- usability testing
- systems engineering methods
- game development with pervasive computing
- cloud computing
- robotics
- virtual reality

Topics

The session features planning, preparation, resource gathering, learning site design, and tips on how to stage learning activities from the initial classes and throughout the course. It also explores the challenges of authentic assessment projects, project evaluations, FERPA compliance, and navigating the issues related to interaction with students in game-like environments.

Figure 1 features a Design Studio Framework for structuring class activities during the creation of projects that apply the course concepts. The tutorial will explain how to use it as well as a rubric designed for staging the activities and assessing the results.

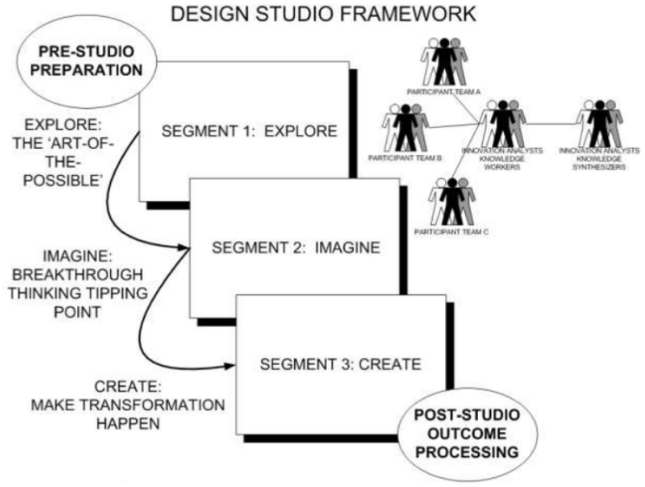


Figure 1: The Design Studio Framework builds on the work by Brown[1]

References

[1] Tim Brown et al. Design thinking. *Harvard business review*, 86(6):84, 2008.

[2] Cynthia Calongne. Cynthia Calongne's presentations [Slides].

[3] Cynthia Calongne. TCC 2021 teaching for the Saints [Slides].

A Comparison of ETL (Extract, Transform, and Load) Tools: Python vs. Microsoft SQL Server Integration Services (SSIS)*

Conference Tutorial

Guangyan Li[1], Mary Donahoo[2], Mario Guimaraes[1]
[1]The Hal and Inge Marcus School of Engineering
[2]Integrated Technology Services
Saint Martin's University, Lacey, WA 98503
{gli,mdonahoo,MGuimaraes}@stmartin.edu

ETL (extract, transform, and load) is a process of gathering large volumes of raw data from multiple (internal and external) sources, transforming the data to meet certain needs, and consolidating data into a single, centralized location such as data warehouse database for analytics or storage. There are many ETL tools available in the data warehousing market, and the choice of ETL tools depends on many factors such as company data needs, existing technology, and budget. Microsoft SQL Server Integration Services (SSIS) is an integrated ETL tool for building enterprise-level data integration and data transformations solutions, and it is a natural choice if SQL Server and other related Microsoft technologies are used. Python is a popular programming language choice for data analytics and data science. Python offers a variety of free ETL tools such as Apache Airflow, Petl and Pandas.

This tutorial aims to expose participants to the basics of ETL process and the role it plays in data warehousing, and help participants choose the right ETL tools for teaching courses such as data warehousing, business intelligence (BI) and business analytics. The complete ETL tasks using both SSIS and Python ETL tools are demonstrated. The pros and cons of using enterprise ETL tools like SSIS and open-source software tools like Python and its libraries are discussed. Given the increasing trend of migrating enterprise data to cloud, other prominent ETL tools, particular those handle cloud data, are also briefly introduced and explained.