The Journal of Computing Sciences in Colleges

Papers of the 30th Annual CCSC Rocky Mountain Conference

October 15th-16th, 2021
Utah Valley University (virtual)
Orem, UT

Baochuan Lu, Editor
Southwest Baptist University

Pam Smallwood, Regional Editor
Regis University

Volume 37, Number 2 **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 Rocky Mountain Conference

Welcome to the 30th annual conference of the Rocky Mountain (RM) Region of the Consortium for Computing Sciences in Colleges. This is our second virtual conference due to COVID-19. The CCSC RM region board members are grateful for the authors, presenters, speakers, attendees, and students participating in this year's conference.

This year we received 17 paper submissions on a variety of topics, of which 11 papers were accepted for presentation in the conference. Multiple reviewers, using a double-blind paper review process, reviewed all submitted papers and tutorials for the conference. The review process resulted in a paper acceptance rate of 64.7%. In addition to the paper presentations, there will be three tutorials/workshops. We truly appreciate the time and effort put forth into the reviewing process by all the reviewers. A special thank you goes to Submission co-chairs Mohamed Lotfy and Karina Assister. Without their dedicated effort, none of this would be possible.

The CCSC RM region board would like to thank our national partners: Turing's Craft, Google for Education, GitHub, the National Science Foundation (NSF), Codio, zyBooks, the National Center for Women Information Technology (NCWIT), TERADATA University Network, Mercury Learning and Information, Mercy College, and the Association for Computing Machinery in-cooperation with SIGCSE. We hope you enjoy the conference and take the opportunity to interact with your colleagues and leave both enthused and motivated. As you plan your scholarly work for the coming year, we invite you to submit a paper, workshop, tutorial, or panel for the 31st CCSC RM region conference, serve as a reviewer or on the CCSC RM region board. Please encourage your colleagues and students to participate in future CCSC RM region conferences.

Mohamed Lotfy and Dan McDonald
Utah Valley University
Conference Chairs

2021 CCSC Rocky Mountain Conference Steering Committee

Karina Assiter, Submission Co-chair ..... Landmark College, VT
Mohamed Lotfy, Submission Co-chair ..... Utah Valley University, UT
Kim Bartholomew and Dan McDonald, Webmasters . Utah Valley University, UT
Sayeed Sajal, Publicity Chair ..... Utah Valley University, UT
Mohamed Lotfy, Conference Co-chair ..... Utah Valley University, UT
Dan McDonald, Conference Co-chair ..... Utah Valley University, UT
Mohamed Lotfy, Program Chair ..... Utah Valley University, UT
Michael Leverington, Student Posters Chair Northern Arizona University, AZ
Aziz Fellah, Student Programming Competition Chair ... Northwest Missouri State University, MO

Regional Board — 2021 CCSC Rocky Mountain Region

Mohamed Lotfy, Board Representative ..... Utah Valley University, UT
Ed Lindoo, Treasurer ..... Regis University, CO
Pam Smallwood, Editor ..... Regis University, CO
Ed Lindoo, Registrar ..... Regis University, CO
Kim Bartholomew, Webmaster ..... Utah Valley University, UT

Reviewers — 2021 CCSC Rocky Mountain Conference

Amin, Mohammad ..... National University, San Diego, CA
Anthony, Barbara ..... Southwestern University, Georgetown, TX
Assiter, Karina ..... Landmark College, Putney, VT
Bandi, Ajay ..... Northwest Missouri State University, Maryville, MO
Blumenthal, Richard ..... Regis University, Denver, CO
Bryce, Renee ..... Utah State University, North Logan, UT
D'Antonio, Lawrence ..... Ramapo College of New Jersey, Mahwah, NJ
Duncan, Denise ..... Regis University, Denver, CO
Fellah, Aziz ..... Northwest Missouri State University, Maryville, MO
Fulton, Steven ..... US Air Force Academy, Colorado Springs, CO
Glass, Michael ..... Valparaiso University, Valparaiso, IN
Harris, Laurie ..... Southern Utah University, Cedar City, UT
Hasan, Mir ..... Austin Peay State University, Clarksville, TN
Leverington, Michael ..... Northern Arizona University, Flagstaff, AZ
Lindoo, Ed ..... Regis University, Denver, CO
Lotfy , Mohamed ..... Utah Valley University, Orem, UT
Mazumdar, Subhasish . New Mexico Institute of Mining and Technology, NM
McDonald, Dan ..... Utah Valley University, Orem, UT
North, Matt ..... Utah Valley University, Orem, UT
Olah, Judit ..... Regis University, Denver, CO
Pinto, Marcus ..... NYC College of Technology, Brooklyn, NY
Sajal, Sayeed ..... Utah Valley University, Orem, UT
Salinas Duron, Daniel ..... Westminster College, Salt Lake City, UT
Smallwood, Pam ..... Regis University, Denver, CO
Taysom, Troy ..... Utah Valley University, Orem, UT

Towards Evaluating Ethical Accountability and Trustworthiness in AI Systems*

Radana Dvorak[1], Hayden Liao[2], Sarah Schibel[2], and Ben Tribelhorn[2]†
[1]School of Technology & Computing
City University of Seattle, Seattle, WA
dvorakradana@CityU.edu
[2]Donald P. Shiley School of Engineering
University of Portland, Portland, OR
{liao21, schibel21, tribelhb}@up.edu

Abstract

Intelligent systems, also referred to as Artificial Intelligence, are rapidly expanding with direct consequences and impacts. Given that the media focuses more on reporting "disasters" involving these systems, and due to both the complexity and often black-box nature of these systems, there is a need for the non-technical audience to understand the impacts given the speed of their adoption.

This paper proposes a process for describing and rating ethical accountability of intelligent systems in order to allow non-experts to contrast and evaluate the trustworthiness of an intelligent system. The output of the application of this framework enables domain experts to assess components of these systems in a public sphere akin to peer review allowing for discussion and discovery of trust issues. Secondly, this output is designed to be understood by the general public, motivating creators of intelligent systems to improve the quality of their systems. Similar methods are utilized, with positive effect, in domains such as public health grades, charity rankings, the equality index, and many

†Corresponding author.

more. This process helps to answer the need for accountability in the deployment of intelligent systems by both public and private entities.

Finally, we report on a survey of Junior Computer Science students about how they would rank various components of a trust evaluation. This preliminary survey helps support our call to action for transparency in intelligent systems and supports the design of our proposed framework.

1 Introduction

Artificial Intelligence (AI) is a well known and wide field of application and research. Colloquial definitions of AI may include fantasy-based connotations. In this work, we define an "AI System" or intelligent system as any computer program that has the ability to impact people through action, inaction, decision, or data collection. With this broad definition, much of the realm of public and privately deployed software would be considered an intelligent system.

Popular media has been reporting incidents of poorly behaved intelligent systems for years, including incidents with self-driving cars, judicial review systems, facial recognition, and job applicant selection to name a few. There is obviously a need for a detailed evaluation of intelligent systems. It is clear that this evaluation is strongly supported by the ACM Code of Ethics as that is based on ensuring the public good is a central concern, specifically calling for moral imperatives of "avoid harm to others," "be fair and take action not to discriminate," "respect the privacy of others," and "foster public awareness and understanding of computing, related technologies, and their consequences."[1] This effort is a step towards these goals.

Given this need for understanding and accountability, this work extends the development of using experts to analyze intelligent systems towards an output that can be understood by a lay audience, specifically the public. In order to ground our selection of important topics, we presented a survey to a class of Junior Computer Science majors in a required course to see what these "experts" value in the evaluation of trust in intelligent systems. Although we were unable to complete the post-survey due to the global pandemic, we present preliminary data to motivate discussion and future work.

2 Background

There is a social contract between members of the public and corporations and governments when interacting. As many of the functions of these entities become enhanced or replaced with intelligent systems, there is a clear need to maintain this expectation. Considerable thought work has been done in this area from Rawls[15] to more recent scholarship that calls for ensuring

that algorithms governing our lives are transparent, fair, and accountable.[13]. We define **trustworthiness** in this article to espouse these ideals, but from the perspective of a citizen of the world as opposed to a rigorously defined metric. As each person may perceive these qualities differently, the authors believe that a focus on all of them will inevitably yield accountability and trustworthiness. Fairness however, may prove illusive as many definitions can be formulated. Researchers are creating precise notions for fairness of decisions processes, e.g.[8, 12, 7]. Ultimately, there are methods for evaluating fairness properties of a system using post-processing such as the work shown on the COMPAS dataset.[3] However, trust is based on an individual's expectations of the social contract not on an absolute measure of fairness. This is further supported by a recent and comprehensive literature review[18] shows that we need to move to an accountability relationship that includes the use, the design, the implementation, and the consequences of algorithmic systems, in addition to the socio-technical process. This work is a proposed component of this important process.

Governments, non-governmental organizations, corporations, and researchers are all invested in moving towards achieving trustworthy AI, each lens brings a unique perspective. Recent efforts from a legal perspective (generative & compliance) were published by the EU.[4] In describing key values, The Alan Turing Institute lists: "fairness, accountability, sustainability and transparency."[9] Major corporations focus on bias, for example IBM lists over 180 different possible human biases that could appear in AI. Additionally, issues of "Accountability, Value Alignment, Explainability, and Fairness" are listed as key concerns by IBM.[5] Recent work by Google focuses on accountability from the perspective of internal audits.[14] Obviously, each entity has a goal in achieving trustworthy AI; this work aims to move towards a simple framework that respects the expected social contract and integrates the various perspectives with a focus on the public good as motivated by the ACM Code of Ethics.

This work leverages prior effort in clarifying intelligent systems, especially ones where machine learning (ML) models are used. The artistry required to build successful models means there is a large value to entities keeping their models and data proprietary. However, work has been evolving into reconstructing models from explanations and behaviors.[10, 17] This suggests that even in proprietary cases, some information can be used to evaluate the intelligent system. Ultimately, there is considerable literature support for improving the explainability of intelligent systems[6] and many methods exist for achieving some level of explanation based on the system components.

Clearly, assessment of individual components of an AI system is a daunting task, but fortunately there are considerable prior efforts. A great effort was recently presented focusing on the data and training of a model to be detailed in a

"Model Card".[11] These model cards include sections such as: Model Details, Intended Use, Factors, Metrics, Evaluation Data, Training Data, Quantitative Analysis, Ethical Considerations, and Caveats and Recommendations for data. Model Cards are still limiting as they require considerable technical knowledge to understand. Within the data, prior work has investigated trade-offs in opening proprietary data, and one group proposes an integrated legal-technical approach provided by a third-party public-private data trust designed to balance these competing interests.[19] However, this work assumes that given the complexity of such systems, trust can still be established without direct access to the data (a less desirable case to be sure). We believe that trust can be gained by knowing some limited information about the data used in training the model similar to the level of information contained in the aforementioned model cards.

When applying these models, many deployments of models are not for end users affected by the model but rather for machine learning engineers, who use explainability to debug the model itself. Therefore there is a predicable gap between explainability in practice and the goal of transparency, since explanations primarily serve internal stakeholders rather than external ones.[2] Some of this prior work is very comprehensive, including *Explainability Fact Sheets*[16], but with a focus on the developers it misses an important facet of accessibility to those impacted by intelligent systems.. This work addresses the need for a external analysis and explanation of these intelligent systems to the public and in a non-technical format that can be understood by a lay audience.

Finally, this work is unique in its centering on the edge between domain experts and the non-expert public. Similar to the global consensus around the handling of dangerous materials through the dissemination of Safety Data Sheets (SDS)[1] which are available to employees and users of dangerous chemicals, AI's impact is equally dangerous, if not so immediate or visible. This work does not require a formal definition of fairness, but investigates the variable levels of trust based on an expectation within the social contract.

3 Methods

Based on an evolving consensus described by researchers, including publicly disclosed efforts by government agencies and corporations, we propose to evaluate intelligent systems under the following categories which are well supported by prior research: 1) Intent and Limitations, 2) Data, 3) Explainability, 4) Safety & Robustness, 5) Auditability and Accountability. Within these categories a series of questions for expert analysis are offered (see Tables 1-2). No specific

[1]The SDS format is based on the UN's Globally Harmonized System of Classification and Labelling of Chemicals (GHS).

scoring is offered as the domain of application would require weighting each question differently. However, for a non-expert, a short analysis from each question would allow them to understand and contrast the trustworthiness of intelligent systems. What follows is a brief explanation of each category in terms of the motivation for inclusion with related question numbers in parentheses. The results section includes commentary on the application of the questions for specific real world examples.

Intent and Limitations: The first set of questions seeks to inform the context of the system to verify that it has been appropriately designed for its actual use. This includes questions to verify that the system shows respect for human autonomy (1.1). It also looks for clear uses including legal compliance (1.2-1.4), and transparency in application (1.5-1.6). Finally, if the system includes technology or code re-use that should be acknowledged (1.7). Ultimately, some technology firms are leading the argument that systems should include details of their development, deployment, and maintenance so they can be audited throughout their software life cycle. Note that a negative score is appropriate for failures to disclose. So an answer of Unable to Ascertain (UTA) is considered as bad as purposeful obfuscation. Additionally, we do not address the issue that a system's legality may vary by jurisdiction.

Data: Data is the linchpin to many intelligent systems as the resulting models created by machine learning algorithms are determined by the input data, which can be drastically impacted by data collection (2.1-2.2), data processing (2.3), and the model design (2.4). Much of the prior work in this area focuses on the impacts of these details on the applications of machine learning. The focus on these questions is on achieving a level of equity and fairness which requires that training data and ML models be, mostly, free of bias to avoid unfair treatment of specific groups.

Explainability: Both users and developers need to understand intelligent systems, but often from different perspectives. Users, as the source of trust, need to believe that they understand a certain amount of the behavior of an intelligent system. This relates to the personification of AI which implies an expected code of conduct of these systems. Often some form of "fairness" is the expectation. We focus on the behavior compared to humans (3.3) and efforts to analyze the system for bias (3.1-3.2). These questions are based on a practical consensus that fairness is too difficult to specify to the satisfaction of all stakeholders, so by analyzing systems on the axis of human biases, we can place intelligent systems in context with the human-based processes they augment or replace.

Table 1: A framework for evaluating ethical accountability and trustworthiness of an intelligent systems

No.	Analysis Question	Positive Score	Negative Score	Partial Score
1.1	Can the user opt out of the AI's functionality?	At any time	No	Extra steps to opt out
1.2	Is the AI legal?	Yes	No or UTA	Contested
1.2	Are the use cases clearly specified?	Yes	No or UTA	
1.3	Can outcomes be challenged?	Yes	No	Some cases
1.4	Are limitations specified?	Yes	No	
1.5	Is any part of the decision-making process disclosed?	Yes	No	
1.6	Can the steps of the decision-making process be identified and scrutinized separately?	Yes	No	
1.7	If this AI replaced another, are issues from the prior system addressed and solved?	Yes	No	Partially addressed
2.1	Is the origin of the data disclosed?	Yes	No	
2.2	Is the data collection process disclosed?	Yes	No	
2.3	If the data was cleaned, is the cleaning process or processes disclosed? OR If the data was not cleaned, is there justification?	Yes	No	Partially
2.4	Is a model disclosed or a model information sheet completed to describe the basis for the algorithm?	Yes	No	Partially
3.1	How biased are the AI system's outcomes?	Acceptable or <= humans	Unacknowledged or > humans	
3.2	Is the AI using any biased data?	No	Yes or undisclosed	
3.3	How does the accuracy of the outcomes compare to humans or human-based systems?	Better	Worse or undisclosed	Comparable

Table 2: Continued: A framework for evaluating ethical accountability and trustworthiness of an intelligent systems

No.	Analysis Question	Positive	Negative	Partial
4.1	Is there a process to identify failures? (i.e. alert a human or halt the AI)?	Yes	No	
4.2	Is the safety process in real time or frequent enough for the application?	Yes	No	UTA
4.3	Is there a process in place to identify security breaches?	Yes	No or undisclosed	
4.4	Is the process to address security breaches in real time or frequent enough?	Yes	No	UTA
5.1	Is there a process in place to ensure AI training data continues to meet or exceed the standards set by the initial product?	Yes	No	System is too young to evaluate
5.2	Is there a process in place to ensure AI outcomes continues to meet or exceed the standards set by the initial product?	Yes	No	System is too young to evaluate

Table 3: Survey results, n=17 respondents. Values are from a Likert scale where 1-5 is {Not at all, Very little, To a small extent, To a moderate extent, To a great extent}.

Question	Mean	Std. Dev.
To what extent do you feel comfortable interacting with an AI system?	3.59	0.94
To what extent do you trust an AI system if you know about its purpose?	3.47	0.62
To what extent do you trust an AI system if you know about its data and design?	3.76	0.83
To what extent do you trust an AI system if you know about its fairness?	3.44	1.15
To what extent do you trust an AI system if you know about its safety?	3.67	0.98
To what extent do you trust an AI system if you know about its accountability?	3.56	0.96

Safety & Robustness: Companies are very highly focused on this area while governments seem to place less weight on it due to the nature of their "customers" being often unaware or unable to opt out. Ultimately, high quality software is something users demand more and more. The system needs to have a path for recovery from failures (4.1-4.2) and be resistant to tampering and the data secure against being compromised (4.3-4.4).

Auditability and Accountability: Something that we propose is to generate trust by establishing processes to continually observe the intelligent system after deployment (5.1-5.2). Especially as modern intelligent systems continuously collect data and use that new data for future decisions or actions, this process seems critical to building ongoing trust. As efforts from governments show, public transparency of processes will be required to generate trust.

4 Results

The authors chose to demonstrate this framework with a simple unweighted scoring of percent of questions answered in the affirmative (with half credit applied for partial scores). Future work should focus on appropriately weighting the questions, however this limited methodology produces a notable result. The authors used publicly available information as of March, 2020.

We choose to investigate two competitors in the area of self-driving cars as this is a very current technology of concern to the public given its rapid evolution and frequency of mention in the news and periodicals. The two most referenced companies were Tesla and Waymo, so we completed this questionnaire for each. With this simple scoring, and some discussion amongst the authors, we arrived at an unweighted score of 62.5% for Tesla's intelligent system and 85% for Waymo's. Weighting each question equally the reader might think of this as an initial trustworthiness score based on a checklist. What is most notable, the authors were surprised by this difference between the companies, and as part of the process to research them found that we gained trust (independently) in the technological system produced by Waymo and in contrast to the result we found for Tesla, the authors did not increase their baseline trust in that technological system. We believe this shift in perception (trust) was caused by the extra clarity and transparency in the presentation of the information on the system detailed by Waymo. This warrants additional studies to confirm a change in trust.

4.1 Educational Intervention and Expert Survey

Each domain of application might weight various components of this framework differently. In order to address the importance of various areas of this frame-

work, we surveyed a class of junior computer science students in a required course at the University of Portland. These students could be considered minor experts given their training in computer science, although they are not domain experts in machine learning. The authors intended to report a pre- and post-survey looking at the change in the perceptions of trust in these students to inspire initial weighting of the framework. However, given the global pandemic and switching to online teaching, the intervention, described briefly below, was incomplete. The results we report are only from the pre-survey.

The educational intervention was focused on exposing the students to the ethical issues surrounding AI within the business and professional context. This work would have included attempting to define fairness in intelligent systems, analyzing their own intelligent systems from the course project, and creating a policy document relating to the intelligent system. Showing differences in perception after this exposure could suggest a need for the creators of intelligent systems to educate their users more to build trust. However, as we cannot support this claim, we offer a sample of computing expert opinions on general perceptions of trust.

Table 3 shows the survey results for six questions on disclosure about "AI Systems." The small survey of 17 students shows that the average trust is small to moderate when some information is known. In general, as one might expect, Computer Science students are comfortable interacting with intelligent systems. The variance in standard deviation for the questions hints that some of these areas are less clear to the students. Based on qualitative responses, this shows that terms like fairness are the least clear and subject to many assumptions. Contrast this to the lower variance in responses to the question on purpose, suggesting for experts who create software this is a clearer component to understand. It is possible that a wider sample of the public would not show as much agreement.

Finally, we asked the students to rank twelve terms by their importance for having trust in an AI system (personally). Using a Borda count to combine the rankings for 15 respondents, we generate the following ranking from highest to lowest: 1) Intent, 2) Safety, 3) Data transparency, 4) Accountability, 5) Data model (system design), 6) Respect for human autonomy, 7) Explainability, 8) Legal Compliance, 9) Fairness, 10) Data processing, 11) Limitations, 12) Auditability.

What is particularly notable, these software developers ranked auditability the lowest, which suggests that without an imposed process this is unlikely to be well implemented at the outset for new intelligent systems. Also notable, fairness ranks as lower importance, presumably given the difficulty in both specification and evaluation. With the number one ranked element being intent, we posit that the creators and distributors of intelligent systems can

gain the most trust by clearly educating users of the software's intent, this is covered by our framework in question 1.2. Ultimately, our framework covers each of these issues, so this kind of data could help inform the development of a weighting for the questions. Given the small survey size, the authors prefer to leave this as future work.

5 Discussion and Future Work

Using this framework to analyze intelligent systems should be grounded more deeply in society's perspective. Specifically, the weighting of each category should reflect the needs of society and the evolving failures of AI. The authors believe that there is room for non-governmental entities to engage with this need. In other fields, non-profits work towards similar goals in transparency and improvement; with this field's rapid growth it calls out for a similar intervention. This framework is a place to begin third party review of the systems that will continue to impact and shape our world.

The authors have presented this framework as a model checklist for creators, designers, developers, and owners of intelligent systems that the public could reference to assist the trust building process. Clearly described systems that score well qualitatively on this survey are ones that inspire more trust. This framework is specifically shortened for accessibility to a lay audience and grounded in the expectations of the social contract. This work helps to reframe the need for the components of trust including fairness and explainability around those impacted by intelligent systems. Our framework is unique in its focus on the user rather than the creators or owners of intelligent systems. Ultimately, this helps reinforce the social-contract.

Acknowledgements

Thanks to Daniel McGinty and the Dundon-Berchtold Institute for financial support. Thanks to Dr. Martin Cenek for his thoughtful discussions.

For additional information or a copy of the IRB-approved survey presented to students please contact the corresponding author.

References

[1] ACM. ACM Code of Ethics and Professional Conduct, 2020.

[2] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of*

the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Ran Canetti, Aloni Cohen, Nishanth Dikkala, Govind Ramnarayan, Sarah Scheffler, and Adam Smith. From Soft Classifiers to Hard Decisions: How fair can we be? In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, FAT* '19, New York, NY, USA, 2019. Association for Computing Machinery.

[4] European Commission: High-Level Expert Group on Artificial Intelligence. Ethics Guidelines for Trustworthy AI, 2019.

[5] Brian Goehring, Francesca Rossi, and Dave Zaharchuk. Advancing AI ethics beyond compliance. Technical report, IBM, 2020.

[6] Leif Hancox-Li. Robustness in machine learning explanations: Does it matter? In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 640–647, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Galen Harrison, Julia Hanson, Christine Jacinto, Julio Ramirez, and Blase Ur. An empirical study on the perceived fairness of realistic, imperfect machine learning models. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 392–402, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent Trade-Offs in the Fair Determination of Risk Scores. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:23, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[9] David Leslie. Understanding artificial intelligence ethics and safety. Technical report, The Alan Turing Institute, Public Policy Programme, 2019.

[10] Smitha Milli, Ludwig Schmidt, Anca D. Dragan, and Moritz Hardt. Model Reconstruction from Model Explanations. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, FAT* '19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. In *Proceedings of the*

2019 Conference on Fairness, Accountability, and Transparency, FAT* '19, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5680–5689. Curran Associates, Inc., 2017.

[13] Iyad Rahwan. Society-in-the-loop: programming the algorithmic social contract. *Ethics Inf Technol*, 20:5–14, 2018.

[14] Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 33–44, New York, NY, USA, 2020. Association for Computing Machinery.

[15] J. Rawls. *A theory of justice.* Cambridge: Harvard University Press, 1971.

[16] Kacper Sokol and Peter Flach. Explainability fact sheets: A framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 56–67, New York, NY, USA, 2020. Association for Computing Machinery.

[17] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 601–618, USA, 2016. USENIX Association.

[18] Maranke Wieringa. What to account for when accounting for algorithms: A systematic literature review on algorithmic accountability. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 1–18, New York, NY, USA, 2020. Association for Computing Machinery.

[19] Meg Young, Luke Rodriguez, Emily Keller, Feiyang Sun, Boyang Sa, Jan Whittington, and Bill Howe. Beyond Open vs. Closed: Balancing Individual Privacy and Public Accountability in Data Sharing. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, FAT* '19, New York, NY, USA, 2019. Association for Computing Machinery.

Using the UCSC Genome Browser in a Database Course*

Barbara M. Anthony
Department of Mathematics and Computer Science
Southwestern University
Georgetown, TX 78628
anthonyb@southwestern.edu

Abstract

Students can benefit greatly from working with real databases in their first Database course. A database for a university is a common textbook example, in part due to its familiarity, but privacy and other considerations typically preclude course access to this and many other large, meaningful databases. This paper reports on two semesters' experience using the University of California Santa Cruz Genome Browser [6] in a Database course, allowing mid-level computer science undergraduates to gain hands-on experience with a large real-world database. Anonymous survey feedback from students in both semesters was positive for both engagement and increased knowledge. The activity described within can easily be adopted by others, requires no software installation, and can be adapted to the desired length and difficulty level.

1 Introduction

Undergraduate database textbooks (e.g. [7, 10]) commonly provide examples that they believe will be familiar or understandable to their audience, such as the prevalent university example. While textbook examples go beyond that, many of the fully fleshed out instances provided are small, highly contrived, simplistic in some form, or fail to generate enthusiasm among students. Though

texts often provide a discussion of implications for large data and various real-world scenarios, students can leave a Database course feeling like they never worked with a database that they would encounter 'in the wild'.

The lack of the use of large real-world databases in a classroom setting can stem in part from the challenges that faculty can face in gaining easy access to such a database for a variety of feasibility and logistical reasons. Data privacy, IT concerns or limitations on software availability, and the need to keep assignments with a particular scope are some of the anecdotal reasons that have been shared for relying on small or simplified databases.

The University of California Santa Cruz (UCSC) Genome Browser [6] available at `https://genome.ucsc.edu/` provides an easy solution to many of the hesitations. For over two decades, the UCSC Genome Browser has provided thousands of users "a mature web tool for rapid and reliable display of any requested portion of the genome at any scale" [3]. With daily hits from the early years exceeding 50,000 [3], and ongoing maintenance and updates including COVID-19 human annotations and the SARS-CoV-2 viral genome [6], it provides a large, meaningful database for exploration.

The author was introduced to the Genome Browser in a 2017 workshop aimed at biologists at her institution. In Spring 2018, when the author first used the Genome Browser in their Database course, there was a SIGCSE workshop on Introducing Bioinformatics Algorithms in CS courses [5] aimed at CS1/CS2/Algorithms courses; the associated handout [4] illustrates the use of the Genome Browser visualization tool only. A paper from the same technical symposium reports on the use of the Genome Browser to bring computational thinking into an introductory biology course [8], again using the web interface. The author is unaware, either anecdotally or in the literature, of other educators using the Genome Browser in a Database course or with direct MySQL interaction. This paper reports on two semesters of such usage in a Database course, in ways that can be easily adopted and extended by other educators.

2 The Course and the Genome Browser Lab Activity

There is a single Database course at the author's institution, an elective typically offered every two years, with a prerequisite of the CS II - Data Structures course in Java. The enrollment is predominantly computer science majors but also some minors and other majors, ranging from sophomores to seniors. Accordingly, their CS experience varies greatly, as does their biology background, with many not having taken a biology course since high school.

Normally one of the two weekly class periods is dedicated to hands-on lab activities designed to allow students to practice skills they are learning in the course. This lab is done in the early weeks of the semester, before students

have become proficient with SQL. Because Google Cloud Platform is used more extensively later in the course, Google Cloud Shell (`https://cloud.google.com/shell/`) is used to interact with the UCSC Genome Browser.[1] Though other options are available, a convenience of Google Cloud Shell is that all activities can be done in the browser from a variety of devices with no software installation required. As an introductory lab, some of the goals include helping students feel comfortable no matter their level of prior experience, encouraging the exploration of connections both within and beyond computer science, executing MySQL commands on the shell, and scratching the surface of what can be done with a large real-world database, leaving opportunities for additional exploration.

The activity guides students through a mixture of precise and open-ended questions, a combination that allows many students to work at their level while executing commands, discovering new resources, making connections, and drawing insights. After a quote from Adleman [1] highlighting the interconnection between biology and computer science to help contextualize the activity, students are provided numerous specific links to pages within the Genome Browser domain along with direction for some guided exploration. They review the provided abstract from a recent version of the Genome Browser's preferred citation for the data (see `https://genome.ucsc.edu/cite.html`); for the Spring 2018 course offering described, that was the 2017 update [11]. Though only a single paragraph, the abstract brings in terminology familiar to many computer scientists including phrases such as *open source platform*, *command-line utilities*, and *mirror site*, as well as a lexicon more familiar to biologists, such as *genome assemblies* and *long-range chromatin interaction pairs*. Students pick three least-familiar CS terms to define, providing details about any sources used. The activity is then repeated with biology terms.

After some in-class discussion of these terms and an overview of the Genome Browser, students are pointed to the source code [2]. Since git is not used in first-year courses at Southwestern, students are given guidance about how to figure out what languages are used in the Genome Browser and in what proportion. The author has found this exercise beneficial in motivating students to value a variety of languages, especially those who would have assumed Python was a larger component. (Note that language percentages vary from semester to semester as the code changes; telling students that explicitly may promote academic integrity.)

Instructions about public access to the MySQL database for the Genome Browser are then discussed, including the details from the FAQ about the

[1] The author is not affiliated with Google (nor UCSC nor any school in the UC system), but does get credits from the Google Cloud Teaching & Learning credits program (`https://cloud.google.com/edu/faculty`) widely available to United States college faculty.

allowable hits per day. The basic lab described within easily stays within those restrictions, with students connecting to the MySQL server running on the UCSC site, having no need to compile or run anything locally. Guidance about "How to: create a partial UCSC genome MySQL database" [9] is provided for reference should it be desirable for future exploration.

Students are instructed to connect to the MySQL server with the provided instructions, which include a flag of `-A` that is typically unfamiliar to even students with more command-line experience. They are asked to determine what that flag means and consider why it might be used here. They then execute the `use hg19;` command, reporting what happens and what they can learn about hg19 from the browser. After next executing `show tables;` students are prompted to find, for example, the row *right after wgRNA*, and report what it is, looking up any acronyms or unfamiliar terms encountered. A phrasing of the italicized form allows the lab to be slightly modified each semester to accomplish similar goals but yield different results.

Next, students run the code in Listing 1, often their first encounter with MySQL queries. They then consider how the results align with what they recall (or have looked up) about the number of human chromosomes. They also deduce what they can about the query syntax itself, and try variants to see if their hypotheses seem to hold, often quickly learning some of the idiosyncrasies of MySQL as compared to other programming languages with which they are familiar, including the extent to which case matters.

Listing 1: MySQL query to provide chromosomes of a given format

```
SELECT *
FROM chromInfo
WHERE (chrom LIKE 'chr_' OR chrom LIKE 'chr__')
ORDER BY size
DESC;
```

Afterwards students explore the Genome Browser's visualization tool, with some guided questions related to one of the chromosomes from the previous query. The activity concludes with open-ended questions about what a computer scientist should keep in mind when designing a tool and database like this, features desirable to end users, and frustrations to attempt to avoid.

3 Results from Classroom Usage

Data was collected from student submissions in Spring 2018 and 2020, with approval from the Southwestern University Institutional Review Board. Due to space limitations, some reported results are from 2018 while others are from 2020. Students who consented to participate completed an anonymous survey

after the lab activity had been completed and submitted. Figure 1 reports on the Spring 2018 responses to the survey questions "How much did the activity increase your knowledge of {biology, databases}?", with 1 representing none, 3 moderately, and 5 massively. When viewed individually, all but two students reported an equal or greater increase in their database knowledge than their biology knowledge from the activity, with 16 reporting one step higher for databases than biology. One student who reported a larger increase for biology made a notation on the database knowledge portion that "It more secured the things we have learned." A majority (19 of the 26 students) had not taken a college biology course. Breaking down the results by gender (17 male, 7 female, 1 nonbinary, 1 no answer) did not provide any particular insight.

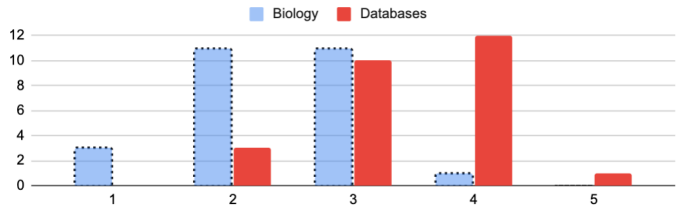


Figure 1: Reported increases in biology and database knowledge from Spring 2018 students after the lab activity.

The survey also asked: *What aspects, if any, of this activity would you retain for the next time this course is offered? Why?* Almost all Spring 2018 responses would keep the activity, with one student saying "Using a large and relevant database from the real world ... helped me stay engaged with the material. I'd like to see this database extended to other projects." Perhaps the most critical comment was from a student with no reported prior biology experience (high school or otherwise), who wanted "a completion of the assignment on a different subject so we know what we're doing." Other suggested changes recommended in the anonymous survey were generally supportive of the activity, requesting "more detailed instructions for how to connect to the MySQL server," additional and more complicated queries, or an optional video on the genome to gain background knowledge on the biology aspects.

Figure 2 highlights commonalities in student responses to what computer scientists should keep in mind when designing a tool like the Genome Browser. Popular themes included efficient access to the data given its size, and ensuring that all users were able to understand relevant terminology.

Figure 2: Word cloud illustrating commonalities in what students in the Spring 2020 course indicated computer scientists should keep in mind when designing a tool like the Genome Browser. Larger words occurred more often.

4 Conclusion

Faculty need not be biology experts to bring a biology database into a computer science course and use it effectively. Using the UCSC Genome Browser serves as a good reminder to faculty of how students may often feel in the classroom, with some knowledge but still much to learn. In a course where students enter with a wide range of prior experience and preparation, activities like this can encourage a cooperative and collaborative atmosphere. Student engagement with the activity and responses to an anonymous survey in two semesters of classroom usage suggest that it was overall effective and engaging.

The activities described within are fully reproducible by others, requiring only a web browser and a means of establishing a connection to the Genome Browser. Though the queries described within are of an introductory nature, instructors can easily customize them to their desired length and difficulty level. A variety of activities, ranging from highly structured to open-ended are possible, including allowing for semester-long student-directed projects, perhaps with collaborators in biology or other departments on campus. In the coming semesters, students and faculty alike may be interested in the SARS-CoV-2 viral genome and other coronavirus-related explorations.

Though the author found it easy enough to experiment with the use of the UCSC Genome Browser, had smooth experiences in the classroom, and recommends its use to others, a caution remains that like any actively maintained site, the content can and will change. For example, the paper abstracts change for the various updates of the database as published in *Nucleic Acids Research* so instructors are advised to ensure clear guidance is given to students about what abstract to read, or to provide the text explicitly for students. Likewise, while the UCSC links have largely remained stable, it is worthwhile reviewing

those each semester. However, overall the changing nature is beneficial, can lead to larger conversations about the impacts of such change, and highlights that the interactions are in fact with a real-world database.

References

[1] L. M. Adleman. Computing with DNA. *Scientific American*, 279(2):54–61, Aug 1998.

[2] W. J. Kent and other contributors. UCSC genome browser source tree. `https://github.com/ucscGenomeBrowser/kent`. Accessed 2021-05-24.

[3] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler. The human genome browser at UCSC. *Genome research*, 12(6):996–1006, 2002.

[4] S. Khuri. Hands-on two, Investigating inherited diseases. `http://www.cs.sjsu.edu/~khuri/SIGCSE_2018/Trans_Trans/SIGCSE_2018_HandsOn_Two_InDisease.pdf`. Accessed 2021-05-31.

[5] S. Khuri. Introducing bioinformatics algorithms in computer science courses: (abstract only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 1054, 2018.

[6] J. Navarro Gonzalez, A. S. Zweig, M. L. Speir, D. Schmelter, K. R. Rosenbloom, B. J. Raney, C. C. Powell, L. R. Nassar, N. D. Maulding, C. M. Lee, B. T. Lee, A. S. Hinrichs, A. C. Fyfe, J. D. Fernandes, M. Diekhans, H. Clawson, J. Casper, A. Benet-Pagès, G. P. Barber, D. Haussler, R. M. Kuhn, M. Haeussler, and W. J. Kent. The UCSC genome browser database: 2021 update. *Nucleic Acids Research*, 49(D1):D1046–D1057, 2021.

[7] C. M. Ricardo and S. D. Urban. *Databases Illuminated*. Jones and Bartlett Publishers, Inc., 3rd edition, 2015.

[8] A. Ritz. Programming the central dogma: An integrated unit on computer science and molecular biology concepts. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 239–244, 2018.

[9] N. Saunders. How to: create a partial UCSC genome MySQL database. `https://nsaunders.wordpress.com/2011/05/18/how-to-create-a-partial-ucsc-genome-mysql-database/`. Accessed 2021-05-24.

[10] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database system concepts*. McGraw-Hill Education, 7th edition, 2020.

[11] C. Tyner, G. P. Barber, J. Casper, H. Clawson, M. Diekhans, C. Eisenhart, C. M. Fischer, D. Gibson, J. Navarro Gonzalez, L. Guruvadoo, M. Haeussler, S. Heitner, A. S. Hinrichs, D. Karolchik, B. T. Lee, C. M. Lee, P. Nejad, B. J. Raney, K. R. Rosenbloom, M. L. Speir, C. Villarreal, J. Vivian, A. S. Zweig, D. Haussler, R. M. Kuhn, and W. J. Kent. The UCSC Genome Browser database: 2017 update. *Nucleic Acids Research*, 45(D1):D626–D634, 2017.

Anticipating Change: Pre-service Elementary Teachers Response to the Challenge of Incorporating Computer Science Activities into Their Future Classrooms*

Cynthia L. Stenger[1], *James A. Jerkins*[2], *Janet T. Jenkins*[2], *and Mark G. Terwilliger*[2]
[1]*Mathematics Department*
[2]*Computer Science and Information Systems Department*
University of North Alabama
Florence, AL 35632
{clstenger,jajerkins,jltruitt,mterwilliger}@una.edu

Abstract

It is a widely held belief among computer science researchers that studying computer science enhances critical thinking, problem solving, and creativity; and it should be a part of the K-12 classrooms. Our home state has joined other states in mandating the latter. Issues of curriculum development, teacher preparation, pre-service and in-service training, funding and logistics, are all important. Over several years, we have developed a novel approach to using computer programming to explicitly teach mathematical generalization and abstraction. In this study, we applied this instructional model to pre-service elementary teachers (PSETs). PSETs participated in four days of explicit instruction where students wrote mini programs designed to push them towards generalization of statistics concepts found in the elementary classroom. Along with instruction, PSETs were given information regarding the important

contributions of computer science and abstraction in their future elementary classrooms. Results showed that PSETs did show improvement in conceptual understanding and also showed that some of the teachers were initially unconvinced that the curriculum was relevant to them or to their future classrooms. The instruction improved PSET attitude and showed that training to meet new computer science (CS) standards in K-6 can and should address this reluctance.

1 INTRODUCTION

A growing body of research supports the conclusion that studying computer science teaches critical thinking and problem-solving skills, creativity, and working collaboratively and that computer science should be employed in an integrated curriculum[4]. George Forsyth, credited by Donald Knuth as founding the discipline of computer science, wrote in 1968 that "The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third"[10, 11]. In one state, the state department of education has formally adopted a computer science and digital literacy course of study that begins in kindergarten and goes through 12th grade [1]. This standard requires computer science topics to be incorporated into the high school curriculum starting in the fall of 2020, middle school in 2021, and finally elementary school and kindergarten in 2022. Additionally, according to code.org, 93% of parents want their child's school to teach computer science and a Google/Gallup survey of that state's school principals indicated that 69% think computer science is just as or more important than required core classes[2].

Given what is a preponderance of both strong beliefs and research results supporting the importance of integrating computer science into K-12 curriculum, there is a concerted effort by education administrators, educational leaders, and teacher professional development programs to infuse CS topics into the K-12 curriculum. Many pre-service teacher preparation programs are also engaged in addressing this issue. However, several barriers exist to the widespread adoption of computer science topics in K-12, including teacher CS knowledge, the belief that other topics (notably, reading and mathematics) will suffer as a result, and lagging financial support for new educational initiatives. One barrier that is less frequently mentioned is that teachers are often reluctant to utilize computer science concepts or practices in their classroom[7, 3, 6].

As a group of mathematics and computer science education researchers, we have seen hesitation to incorporating computing into mathematics lessons in our PD work with K-12 teachers. Our experience has been illustrative of the widely observed pattern where teachers are often initially hesitant, but their

students are eager. We designed a PD approach where teachers work with their students to demonstrate the efficacy of the instructional model. They learn along with their students to employ an instructional strategy with computer programming activities that provides explicit instruction that promotes development of abstract representations of specified mathematical concepts. In addition to utilizing our instructional model with in-service teachers, we have also tested it with undergraduate computer science students and pre-service secondary education math students[13, 9, 8]. We have recently begun investigating the effectiveness of our model with PSETs[9]. Whether elementary students are able to engage in authentic computer science activities and if it is beneficial to them is an active area of research. In 2017, Rich et al. investigated what we can expect most elementary school students to accomplish in terms of computational learning trajectories[12]. Their study starts with empirical data and builds and establishes the relative difficulty of various computer science concepts and detailed learning goals for elementary age students. Their research shows evidence that the computer science concepts of sequence, repetition, and conditionals can be incorporated into elementary curricula effectively and that students can benefit from the approach.

Given that there is widespread support for incorporating computer science into the K-12 curriculum, it is important that PSETs believe that computer science is relevant or useful to them. Guzdial notes in [5] that he observed at the 2020 CUE.NEXT workshop that "lots of education faculty think CS is going to go away". While presenting our Instructional Model (IM) that guides students through writing mini computer programs in order to push them to make generalizations over statistics concepts in the elementary classroom, we observed that many of the PSETs were not initially convinced that computer science topics would benefit them. We disclosed that state departments of education were mandating it, that it had proven pedagogical advantages, and that their career prospects were improved by incorporating computer science activities into their teaching. In this paper, we present data collected throughout our study showing evidence PSETs were initially reluctant to incorporate programming in their future classes and that this type of lesson can improve these attitudes for future teachers.

2 METHODOLOGY

This study took place in an undergraduate PSET statistics course with a total of 19 students at a regional four-year university. There were four days of instruction using our IM, that is an explicit method for teaching abstraction and generalization through writing mini computer programs. PSETs explored statistics concepts that are part of the course of study in the elementary class-

room. The students were introduced to Python and shown how to use variables, print statements, and a while loop. The objective was to use programming exercises to explore the general expressions associated with the mean of an arithmetic sequence of numbers. Students completed a pre-test, response sheets, and a post-test during the study. Students also completed survey questions to investigate their attitude towards learning to program in the context of exploring a mathematical concept. Free response questions were asked following each lesson to discover the students' reactions to our instructional model. These questions were focused on investigating how students felt about the addition of computer programming in their PSET math course. The following is a description of the lesson activities for days 1-4. On day one, PSETs were informed of the recent change in the state's course of study requiring computer science topics in K-12, the importance of computer science and programming in the world today, and the benefits programming could bring to their job prospects after graduation. Students were given Python code with a simple loop to print out consecutive numbers in a given range. The first range explored was [1..3], using the loop below:

Code:	**Output:**

`i = 1`
`while i <= 3:`
`  print (i)`
`  i = i + 1`
	1
	2
	3

Students were asked to predict the output, consider how the code achieved the output, and experiment with the loop to observe the results. Students were provided debugging assistance. They explored other ranges, such as [0..3], [1..5], and [3..6]. They did this by changing the initial value of i and changing the right operand in the conditional expression to the new terminal value. Students were asked to explain to their peers what was occurring in the program. The code above was modified through interaction between the teachers and students to produce a range of even numbers. For example, [2..8], would produce 2, 4, 6, 8. This range was represented by the following code:

Code:	**Output:**

`i = 1`
`while i <= 4:`
`  print (2 * i)`
`  i = i + 1`
	2
	4
	6
	8

It is notable that students could have achieved the column of even numbers by simply initializing i to 2 and increasing the step size in the statement "i = i + 1" to "i = i + 2" without altering the print statement. The goal, however, was to employ the use of the mathematical general expression of an even number

by observing its behavior in the program. Since we want the students to begin to recognize the notion of an even number as 2n, the general expression shows up in the code as 2*i in this program. After their first programming activity, students were asked to take a blank sheet of paper and draw a representation of what was happening in the program. They were asked to think about what the computer was doing and illustrate their mental model of the computer's actions to produce the output. On the second day, the code from day one to print the range [1..3] was repeated and reviewed. In addition to the review, the students were asked to trace the code to count how many times the loop executed. A counter variable was added to the code to set the stage for computing the mean. The counter variable implemented the students' manual action in the previous code tracing exercise. The code displayed below depicts the loop with the counter added. Students experimented with various ranges for i.

Code:

```
counter = 0
i = 1
while i <= 3:
     print (i)
     counter = counter + 1
     i = i + 1
print("Total loops from counter =", counter)
```

Output:

```
1
2
3
Total loops from counter = 3
```

During the next phase of the lesson, the students were asked to use the Python interactive shell to execute simple, one-line statements and search for numeric patterns:

```
>>> (1 + 2) / 2
1.5
>>> (1 + 2 + 3) / 3
2.0
>>> (1 + 2 + 3 + 4) / 4
2.5
>>> (1 + 2 + 3 + 4 + 5) / 5
3.0
```

Students continued this process all the way up to 10. They noted patterns they discovered, discussed the patterns with peers, and shared them with the group. Next, students attempted to generalize the patterns they observed for consecutive integers in the previous exercise. They modified their program to use the counter and sum to calculate the mean.

The modified code is illustrated here:

Code:

```
counter = 0
sum = 0
i = 1
while i <= 3:
     print (i)
     sum = sum + i
     counter = counter + 1
     i = i + 1
print("sum is ", sum)
print("mean is ", sum / counter)
```

Output:

```
1
2
3
sum is 6
mean is 2
```

The students were asked to modify the previous program to produce the mean of consecutive even integers. The modified program is shown next:

Code:

```
counter = 0
sum = 0
i = 2
while i <= 8:
     print (2 * i)
     sum = sum + (2*i)
     counter = counter + 1
     i = i + 1
print("sum is ", sum)
print("mean is ", sum / counter)
```

Output:

```
4
6
8
10
12
14
16
sum is  70
mean is  10.0
```

On the last two days, students returned to the classroom and summarized what they had observed in their programming activities. They were asked to describe general expressions for consecutive numbers and general expressions for consecutive even numbers. They worked together to relate these general expressions to the code they had written. They spent time writing conjectures regarding the mean of 3 consecutive numbers, writing the conjectures using general expressions, then writing convincing arguments using the general expressions.

For example, students conjectured and proved the following: Let 3 numbers x_1, x_2 and x_3 be consecutive. If we let x_1 be any number a, then $x_2 = a+1$ and $x_3 = a+2$. Summing the 3 numbers we have $x_1 + x_2 + x_3 = a + (a+1) + (a+2) = 3a + 3$. To find the mean of the 3 numbers, we divide by 3 so the mean of x_1, x_2, x_3 is $(3a+3)/3 = 3\ (a+1)\ /\ 3 = a+1$, which is the middle number x_2.

This was followed by a conjecture and convincing argument about the sum of 3 consecutive even numbers like this: Let 3 numbers e_1, e_2 and e_3 be consecutive even numbers. If we let e_1 be any even number, then there exists a number b so that $e_1 = 2b$, then $e_2 = 2b+2$ and $e_3 = (2b+2)+2$. Summing the

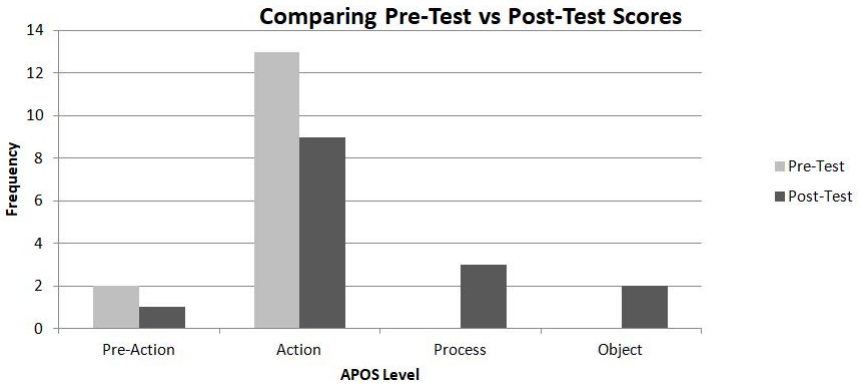
3 even numbers, we have $e_1 + e_2 + e_3 = 2b + (2b+2) + (2b+2)+2 = 6b + 6$. To find the mean of the 3 even numbers we divide by 3 so the mean of e_1, e_2, e_3 is $(6b+6)/3 = 3\ (2b+2)\ /\ 3 = 2b+2$, which is the middle number e_2.

On the final day, students were asked to write about the generalizations they had observed over the course of instruction. They were asked to conjecture and write convincing arguments for the sum of 3 consecutive odd numbers. Finally, they were asked to write a general expression for the mean of any 3 numbers.

3 RESULTS

Our instructional methodology is based on APOS (Action, Process, Object, and Schema are the concept mastery levels) theory, which utilizes a genetic decomposition to analyze mathematical understanding [5]. We developed a genetic decomposition for the mean of an arithmetic sequence and used it to score the student responses. All responses were scored by all four of the researchers. After individual analysis, final scores were assigned to each response based on agreement. When there was not agreement, researchers discussed the score until there was agreement. Additionally, disposition questions were asked to determine how students perceived the programming activities and the call for generalization. There were free response and Likert scale responses.

The pre-test revealed that even though most PSETs could describe the formula for computing the mean, hardly any of them could think about and apply the formula in general terms. For example, student S14 describes computing the mean correctly but when asked to describe the mean of three integers x1, x2, x3, she said "I can't do this because of the letters." It is tempting to say students did not know how to deal with general expressions, but all of them previously passed a pre-calculus algebra course.



As shown above, most students began at the action level as evidenced by attempts to make the pre-test question about numbers. Some students even converted the variables to numbers, x1 to 1, x2 to 2, and so on. Overall, students progressed during this instruction, as five students ended up at either the process or object levels.

PSETs initially expressed frustration with the programming. S3 commented, "The mean was quiet [sic] easy, but figuring where to put things whether it was inside or outside the loop was very difficult." Some also felt it would be difficult to teach programming to elementary students. S11 said, "I believe that it is too difficult for young children to be able to understand." Overall, the lesson addressed and improved these feelings. PSETs began to feel more comfortable as typified by S7 and S9's comments. S7 commented, "I am very unfamiliar with computer programs so the first day was a little confusing for me. My brain tends to work a little slower when it comes to computers and numbers. However, I was getting more comfortable with the program towards the end and looking forward to learning more today." Later S7 added, "I definitely understand the program better on day 2. I think this day was beneficial to me because I was more familiar with Python. Finding how many times it looped was still difficult, but I understood the mean better." S7 modeled the changing sentiment of the class, "I am familiar with these concepts now and I appreciate you taking the time to teach this because it will help me help my future students". Similarly, S9 exemplifies observed improvement, "At first I did not understand the material, but I quickly caught on! I liked how the program printed 3-4 consecutive numbers. I enjoyed the interactive shell to discover patterns of the means and sums. The program worked great."

4 CONCLUSION

The objective of our instructional design used computer programming to push the PSETs to generalize and abstract about the mean of an arithmetic sequence of numbers. Our research has shown that if students write mini programs that iterate over a sequence of numbers to explore a mathematical concept, then students learn to see the general expression in the program and translate or transfer that to the problem under investigation. The overall instructional model was successful, even though the computer programming activities were met with initial resistance. Learning even introductory computer science is no small feat. There are many ways learners of computer science must scale to master concepts. Teaching computer science topics across the K-12 curriculum poses many challenges including teacher preparation, curriculum development, training in-service teachers, and logistical issues such as funding and equipment. This study began as an application of the IM to a new group, PSETs.

Early in the project, however, it was evident that attitudes were impacting student success in programming; so, we addressed them, and they improved. Even though a small number of the PSETs held on to initial beliefs, most of the PSETs showed improved attitudes from their pre-test. To successfully prepare PSETs for new state CS standards, it is beneficial to acknowledge and address these attitudes during teacher training.

References

[1] Alabama State Department of Education. Alabama Course of Study: Digital Literacy and Computer Science. `https://www.alsde.edu/sec/sct/COS/Final\%202018\%20Digital\%20Literacy\%20and\%20Computer\%20Science\%20COS\%205-14-19.pdf`, 2019. Accessed: 2020-05-21.

[2] Code.org. Supporting Computer Science Education in Alabama. `https://advocacy.code.org/`, 2019. Accessed: 2020-05-21.

[3] Mark Guzdial. The role of emotion in computing education, and computing education in primary school: Icer 2017 recap. `https://computinged.wordpress.com/2017/09/01/the-role-of-emotion-in-computing-education-icer-2017-recap/`, 2017. Accessed: 2020-05-21.

[4] Mark Guzdial. Computing education as a foundation for 21st century literacy. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE'19, pages 502–503. Association for Computing Machinery, 2019.

[5] Mark Guzdial. Teaching teachers to offer stem to undergrads. *Commun. ACM*, 63(4):10–11, March 2020.

[6] Mark Guzdial and Briana Morrison. Growing computer science education into a stem education discipline. *Commun. ACM*, 59(11):31–33, October 2016.

[7] International Society for Technology in Education (ISTE). Should we teach computer science in elementary school? Yes. `https://www.iste.org/explore/Point-Counterpoint/Should-we-teach-computer-science-in-elementary-school\%3F-Yes?articleid=216`, 2014. Accessed: 2020-05-21.

[8] Jay L. Jackson, Janet T. Jenkins, James A. Jerkins, Cynthia L. Stenger, and Mark G. Terwilliger. Exploring the Genetic Decomposition of Interior and Exterior Angles of Polygons with the Use of Computer Programming and GeoGebra. In *Proceedings of the 23rd Annual Conference on Research in Undergraduate Mathematics Education (MAA SIGMAA on RUME 2020)*, February 2020.

[9] Jay L. Jackson, Cynthia L. Stenger, James A. Jerkins, and Mark G. Terwilliger. Improving abstraction through python programming in undergraduate computer science and math classes. *Journal of Computing Sciences in Colleges*, 35(2):39–47, October 2019.

[10] Donald E. Knuth. George forsythe and the development of computer science. *Commun. ACM*, 15(8):721–726, August 1972.

[11] Donald E. Knuth. Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4):323–343, 1974.

[12] Kathryn M. Rich, Carla Strickland, T. Andrew Binkowski, Cheryl Moran, and Diana Franklin. K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, ICER '17, pages 182–190. Association for Computing Machinery, 2017.

[13] Cynthia Stenger, Janet Jenkins, James A. Jerkins, and Jessica Stovall. An Explicit Method for Teaching Generalization to Pre-Service Teachers Using Computer Programming. In *Proceedings of the 20th Annual Conference on Research in Undergraduate Mathematics Education (MAA SIGMAA on RUME 2017)*, February 2017.

Walking the Curricular Talk: A Longitudinal Study of Computer Science Departmental Course Requirements*

Richard Blumenthal
Department of Computer and Cyber Sciences
Regis University
Denver, Colorado 80221
rblument@regis.edu

Abstract

A longitudinal study (2014-2021) of the Bachelor of Science in Computer Science degree requirements specified by 380 computer science programs in the United States is presented. Specifically, the study data surveyed the average number of required and elective computer science course credits and the average percentage of fifteen computer science courses required to complete the degree. An analysis of the significant and trending requirements changes between the start and end of the survey is also presented with a discussion or what factors may be accounting for these changes.

1 Introduction

Interest in the relation between desired curriculum, "the *talk*", and the actual courses required by universities to satisfy a computer science degree, "the *walk*", has a long history within undergraduate computer science education. Two articles in the education-focused April, 1964 issue of *Communications of the ACM* nicely exemplify this distinction. In the first, Keenan is motivated by the "desire to understand what courses should properly be given and how they can relate to each other in a well-formed curriculum"[11]. In the second, Atchison

and Hamblen present the results of a survey focused on the computing degrees offered by ninety-three universities in 1963[6]. Despite their differences, the two approaches used in these "historic" articles are motivated by the desire to provide students with an appropriate computing education. More specifically, Keenan's approach stemmed from the position that while "established disciplines can cite course content as a means of defining the substance of the field ... this method is inadequate in a developing field..."[11], which computer science certainly was in 1964. Alternatively, Atchison and Hamblen's focus on curriculum stemmed from the position that "it behooves us all to do whatever we can to see that good sound programs will continue to develop from the already fertile ground" of existing programs[6]. As members of the ACM Curriculum Committee on Computer Science, Atchison, Hamblen, Keenan, and others subsequently contributed to the first published undergraduate computer science curricular recommendations, *Curriculum 68*. These recommendations included both topical and course focused requirements[2][3].

Since *Curriculm 68*, undergraduate computer science curricular recommendations have been published approximately every ten years by the ACM, and then Jointly with the IEEE Computer Society since 1991. Dziallas and Fincher provide a nice overview summarizing how the pedagogic perspective of these recommendations shifted from a primarily course-based expectation to a curricular body-of-knowledge focus[9]. They also provide a characterization of the "committee members' assumptions and perceptions about how a report will be used" including curriculum as prescription, the *talk*, and actual use by universities, the *walk*[9]. Be it the focus on courses suggested in the earlier curricular reports or the body-of-knowledge in the later reports, it is easy to find and track the history of specified recommendations, again the *talk*, in the published literature available on the internet via the ACM Digital Library.

Alternatively, it is not easy to find an historical *walk* of courses required by computer science programs. Knowledge of such course requirements can provide faculty developing curricula and courses within an undergraduate computer science program further context, or "fertile ground", for understanding the current state of undergraduate computer science education, as represented by how other universities are approaching curricular recommendations. Additionally, this context can be used as additional input into the development of the next curricular recommendations, which at the time of this writing are identified as CS'202X[5]. The description of the Core Tier-1, Tier-2, and Elective topics in CS'13 states that curricular committees are well aware of "the ever increasing pressure to grow the core ... and impossibility of doing so within the short-time-frame of an undergraduate degree"[4]. Consequently, an understanding of how universities are addressing requirements can be helpful and utilized, as needed, by university and international curricular committees.

Towards this end, the remainder of this article presents a 2014 to 2021 longitudinal study of the courses and credit hours required by various Bachelor of Science in Computer Science (BSCS) programs within the United States. An analysis of changes over the length of this study is also presented. Specifically, the longitudinal nature of the study makes it possible to test a null hypothesis that assumes the average credit hours required to graduate from a BSCS degree hasn't changed. Furthermore, the data allows testing hypothesized requirement changes in any of the 15 courses from the 2014 study. It would be expected that historically required courses, such as Operating Systems, are required at the same levels, but newer courses, such as Web Development, are increasing (i.e. required by more BSCS programs in 2021 than in 2014).

2 Background

In 2014, a document analysis survey of the course and credit hours required by 388 BSCS programs in the United States (U.S.) was conducted[7][1]. Specifically, credit hour requirements for required and elective computer science courses was collected. These credit hours did not include required non-computing courses, such as mathematics, natural sciences, or undergraduate core requirements. Additionally, information as to whether 15 different computing courses were required to complete the BSCS degree was also collected. These courses included those identified as being part of an initial sequence, beginning with a CS0 or CS1 type course and ending with a CS2 or CS3 course. Additional frequently occurring undergraduate computer science courses were also included in the survey. Likewise, information on required Discrete Structures/Mathematics course requirements was also collected, but not included as part of the required computer science course credits (i.e. they were treated as a mathematics course despite which department offered the course). Relevant results of this analysis are presented later in this article, as part of a comparison between the previous 2014 and the current 2021 survey. The 2014 survey was motivated by the growing body of knowledge in computer science and presented ten design patterns used by BSCS programs to handle this growth within the fixed time constraints associated with completing a degree. These patterns included, for example, combining the topics from two courses into one, such as *Web and Database Development*, creating elective specializations, such as Theory versus Security concentration, and spawning new degrees, such as Software Engineering.

[1]Document analysis is a research procedure for reviewing or evaluating documents[8]

3 Methodology

In May 2021, curricular course and credit hour data was collected via a document analysis survey[8] of the degree requirements for BSCS programs, as specified in 380 (N=380) university catalogs.

3.1 Sample Population

The BSCS programs selected for analysis in 2021 correspond to those analyzed in a the previous 2014 survey[7]. In both surveys, all computer science programs in the U.S. accredited by the Computing Commission of the Accreditation Board for Engineering and Technology (ABET) in 2014 were selected. The U.S. Institute of Educational Science Universities (IESU) list was also used to select additional computer science programs by including in the sample every state university with a name, such as, "The University of Arkansas", "Arkansas State University", and "Arkansas Technical A&M University". Finally, additional public and private programs were randomly selected from the IESU list for inclusion in the sample. The reduced size of the current survey (N=380 vs. N=388) reflects the fact that several universities switched from offering a BSCS degree to other computing degrees including a Bachelor of Arts in Computer Science or Software Engineering, which are not included in the study, or universities that no longer offer a BSCS degree. Also, two universities merged into a single university. The sample is also limited to those universities whose catalogs can be found online. In the 2021 survey, approximately eight BSCS programs were included per state (μ=7.73, σ=4.56).

3.2 Procedure

For each BSCS program in the sample population, the university's published online catalog was used to determine the credit hours for required and elective computer science courses. As the credit hours in this article are based on a semester system, the reported credit hours for universities on a quarter system were converted to semester hours by using the common practice of diving by 1.5 to convert, for example, a 180 quarterly credit hour requirement to 120 semester-based hours[10]. The computer science credit hours include computing-focused courses, even if offered by another department. For example, a required *Database* course offered by a different Computer Information System (CIS) department is included as a computer science course. Required writing courses were not included in the credit hours, unless they were specifically offered by the computing department granting the degree. Additionally, Discrete Structure/Mathematics courses designated as a computer course were counted as mathematical credits. Credit hours associated with required courses

focused on social, ethics, or legal issues, Knowledge Area *Social Issues and Professional Practice* (SP) in CS'13, were only included as computer science credit hours, when offered by the same computing department granting the BSCS degree. Finally, a handful of BSCS programs have switched from using any non-required computing course as an elective to requiring specific computing courses as part of a **required** concentration (i.e. random elective courses cannot be mixed). These required concentration credit hours were included in the required hours, as opposed to in the elective hours.

Next, in order to determine the portion (percentage) of specific courses required to satisfy a BSCS degree, a tally of each required computing course within a degree was also kept. As listed in Table 2, tallies were collected for the fifteen courses reported in the 2014 survey. Generally, the names of courses were used to distinguish them; though, course descriptions were used when a course name could not be used to determine its topical matter for classification. The following distinctions were used when collecting the course tallies:

- CS0 - a course was only considered to be a CS0 course, if it was a prerequisite of a subsequent CS1 programming course.
- CS3 - a course was only considered to be a CS3 course, if it immediately followed a CS2 course and was followed by a subsequent advanced *Algorithms and Analysis* or advanced *Data Structures* course later in the curriculum. Hence, if a program only contained a single non-CS2 data structures and algorithms course, it was not classified as a CS3 course, but instead, as an *Algorithms and Analysis* course.
- Software Engineering - capstone and senior projects that specifically included "software engineering" in the course title or software engineering outcomes in the course description were included in the Software Engineering tally. However, the credit hours were not duplicated in the tallies (i.e, a six-credit SE course counted as a single course in the required percentages, but as six credits in the required computing credit hours).
- Computation Theory - formal language, automaton, or theory courses were only included as a *Computation Theory* course, if the they addressed Turing Machines and decidability results, such as the halting problem.
- SP - social, ethical, and legal focused courses were included in the tally, as long as they were specific to computing and technology, even if offered by a Philosophy department (i.e. they were not included, if they were part of a required core education general ethics course).
- Concentrations - while elective concentrations were included as required credit hours, the courses in these concentrations were not included as required courses in the percentage tallies since a student could select a different concentration with a different set of required courses (hence, specific courses within a required concentration are, in fact, elective) .

4 Results

This section presents the results of the 2021 survey. To facilitate comparison, certain results from the 2014 survey are also reproduced in this section.

4.1 Credit Hours

As shown in Table 1, the average number of required, elective, and total computer science credit hours increased between the 2014 and the 2021 surveys.

Table 1: Average Required Computer Science Credit Hours

Year	Required	Elective	Total
2013	36.88	11.14	47.91
2021	39.63	11.59	50.92

Three independent sample t-tests were conducted to compare the average, M, number of required, elective, and total computer science credit hours between the 2014 and 2021 surveys (i.e., a null hypothesis of $H_0 : M_{2021} = M_{2014}$).

1. There was a significant difference (increase) in the average number of require CS credit hours for the 2014 (M=36.99, SD=8.25) and 2021 (M=39.64, SD=10.71) surveys $t(766)$=4.01, p<.001.
2. There was no significant difference in the elective CS credit hours for the 2014 (M=11.14, SD=6.10) and 2021 (M=11.59, SD=7.4) surveys $t(766)$=0.92, p=.36.
3. There was a significant difference (increase) in the total number of CS credit hours for the 2014 (M=47.91, SD=8.79) and 2021 (M=50.92, SD=9.93) surveys $t(766)$=4.45, p<.001.

These results suggest that the the number of required computer science focused credit hours required to satisfy a BSCS degree has increased during the period from 2014 to 2021, as a result of required computing courses.

4.2 Required Course Percentages

Table 2 give the percentage of courses, P, required by BSCS degrees in the 2021 survey. For comparison, Table 2 also gives the percentage of courses required by BSCS degrees in the 2014 survey (the bold font is explained below).

For each of the courses appearing in the 2014 and 2021 samples (Table 2), a two-population proportion z-test was conducted to compare the percentage of BSCS programs requiring a specific course between the 2014 and 2021 surveys

Table 2: Proportion of Required CS Course Comparison

Course	2014	2021	Course	2014	2021
CS0	.16	.16	Prog. Languages	.63	.69
CS1	.99	.99	Comp. Theory	.46	.40
CS2	.99	.99	Database	.36	.39
CS3 Data Structs.	.65	.46	**Networking**	.25	.32
Alg. Analysis	.61	.86	Web	.10	.09
Computer Arch.	.83	.86	AI	.06	.07
Operating Sys.	.78	.76	HCI/UI	.04	.02
Software Eng.	.64	.73			

with the null hypothesis ($H_0 : P_{2021} = P_{2014}$). The following courses were found to have significant differences at the $p = .001$ and .01 levels and a slight significance at the .05 level (indicated with bold font in Table 2):

1. There was a significant difference (decrease) in the percentage of required CS3 courses for the 2014 (N=61, P=65%) and 2021 (N=179, P=46%) surveys z=4.98, p<.001.
2. There was a significant difference (increase) in the percentage of required *Algorithms and Analysis* courses for the 2014 (N=238, P=61%) and 2021 (N=328, P=86%) surveys z=-7.86, p<.001.
3. There was a significant difference (increase) in the percentage of required *Software Engineering* courses for the 2014 (N=250, P=64%) and 2021 (N=279, P=73%) surveys z=-2.69, p<.01.
4. There was a significant difference (increase) in the percentage of required *Networking* focused courses for the 2014 (N=98, P=25%) and 2021 (N=128, P=32%) surveys z=-2.56, p<.05.

While not statistically significant at the p<.05 level, the *Programming Languages* course percentage showed a strong trending increase and the *Computation Theory* course a slight trending decrease. The specific results are:

1. There was a strong trending difference (increase) in the percentage of required *Programming Languages* courses for the 2014 (N=245, P=63%) and 2021 (N=264, P=69%) surveys z=8.17, p=.06.
2. There was a slight trending difference (decrease) in the percentage of required *Computation Theory* courses for the 2014 (N=177, P=46%) and 2021 (N=151, P=40%) surveys z=1.65, p=.09.

No significant differences were found for the other nine courses in Table 2,

5 Discussion

The availability of course names and descriptions in online university catalogs facilitates the type of document analysis survey presented in this article. The historical nature of course names, along with their descriptions, provides a strong indication of the topics taught within a particular course, which allows a reasonable understanding of the computing material that computer science students are exposed to (on average) in their BSCS degree. Though requiring several weeks of data collection, this document analysis approach balances the additional effort that would be required to gather more in-depth topical information from course syllabi, which are not readily available on the internet. It also trades off against an approach using survey requests to individual departments, where response rates less than the close to 100% of catalog availability would be expected. Despite certain limitations in the survey procedures (see below), the longitudinal data provides a means to evaluate the evolution of BSCS degree requirements over the seven period of the study. Consequently, the credit hour averages and percentages of the fifteen courses required to satisfy the BSCS degree requirements provides an informative view into how computing departments are actually addressing the evolving curricular requirements associated with the *walk* via the courses they require, the *talk*.

As suggested by the survey results, the number of computing courses needed to satisfy a BSCS degree has increased by one three-credit computing course over the seven-year period from 2014 to 2021. Furthermore, this increase resulted from the addition of a named, required computing course specified by the program (instead of an increase in an elective courses that may be selected by a student). This increase suggests that departments may be responding to the increasing body of computer science knowledge by presenting additional curricular material as a new course. Given the premise of the 2014 study, which assumed a fixed number of credit hours, this result is somewhat surprising since it suggests a simpler design pattern than those previously presented. Namely, an increase in coverage of material, as opposed to, for example, adding material to existing courses. While this type of increase is a natural way to handle such an increasing body of knowledge problem, it is important to realize that every credit hour increase within a major results in an equivalent decrease elsewhere in the degree since university credit hours tend to remain fixed or are decreasing[12].

Although the course comparisons given in Table 2 show that decreased requirements occurred in only four required courses, of which one was significant and one trending, there does not appear to be a single course or two that has contributed to the three-credit hour increase in required credits. Instead, the evidence suggests that various courses added by different programs appear to be contributing to this increase. Such a spread of increase across courses can

be accounted for by observing that various institutions are at different initial starting places for courses that have not been part of the historical curricular recommendations, such *Operating Systems*.

Furthermore, the significant decrease in the percentage of required *CS3* focused data structures courses, in conjunction with the significant increase in *Algorithms and Analysis* courses, suggests faculty believe that intermediate and advanced data structures and algorithms including intermediate and advanced complexity analysis, where "intermediatiate" implies beyond arrays and linked lists, are sufficiently covered by two courses beyond the introductory CS1 course, instead of three. More specifically, by coverage of material in the *CS2-Data Structures* and *Algorithms and Complexity* courses. Anecdotally, this is the approach the author's department has moved to, where data structures and elementary complexity analysis are covered in the *CS2 Data Structures* course and intermediate and advanced algorithms and complexity analysis in a subsequent *Algorithms and Analysis* course. For example, topics such as the Master's Theorem and Big-Omega (vs. Big-O) in the advanced course.

It is further hypothesized that the significant increase in required *Software Engineering* courses result from an ongoing shift away from the more theoretical focus of computer science in the past to a practical ability of students to develop software applications. Evidence for this hypothesis is further supported by the decrease trend in the percentage of BSCS degrees requiring a Computation Theory focused course. During the data collection, it was found that many programs have a concentration theory track that allows students to take, for example, either an advanced algorithms or computation theory course, or even courses such as security, whose topics are completely different, even if theoretical in nature.

The recent shift in the Computing Commission of ABET curricular criteria to require "a major project that requires integration and application of knowledge and skills acquired in earlier course work"[1] may also be contributing to the increased percentages in *Software Engineering* courses since many required capstone and senior project courses were found to include software engineering in the course name or software engineering topics in the course description. A similar shift in ABET criteria may also be contributing to the increase in Networking courses since "exposure to networking and communication ... and parallel and distributed processing" are now also required by the ABET criteria[1]. It would be interesting to explore in future research how much the ABET accreditation criteria are contributing to changes in degree requirements, when compared to CS'13 recommendations and non-accredited program requirements.

The lack of increases in Web and AI courses is somewhat surprising since the percentage of programs requiring such courses remains low with no increase as a required course observed. This data suggests these courses are still being treated as elective courses. This is especially true for the Web-focused courses, which represent a relatively new CS area, when compared to the traditional courses of *CS1, CS2, Computer Architecture, and Operating Systems*, which date back to the inception of CS curricular recommendations[3].

There are three course types for which credit hour information, but no tally information, was collected (an unfortunate oversight by the author). The first are introductory type CS0 courses that are **not** a prerequisite of a subsequent CS1 course (prerequisite *CS0* courses were tallied). The second are Digital and Boolean Logic focused courses and the final type are systems programming type courses. It is believe these courses are required by less than 10% of all programs. Clearly, insights provided by the survey are restricted by the course tallies collected. While the longitudinal nature of the current 2021 study was restricted by the courses originally examined in the 2014 study, future surveys should categorize and tally all courses required by a BSCS degree.

Future research could also attempt to further discriminate the types of courses used to satisfy a degree by categorizing similar programs. For example, comparing degrees that require a CS0 or CS3 course against those that do not. It would also be possible to compare external curricular influences, such as, whether the program is ABET accredited versus those programs that are not. Purely external criteria could also be used to compare programs, such as, whether a granting university is private or public. The current data could also be used to determine what courses were added/deleted by a specific program since the non-averaged raw-data for each program contains this information.

It is hoped that the actual course offerings and longitudinal data provided in this article, the *walk*, along with available curricular recommendations, the *talk*, helps to provide better insight into understanding what constituents a computer science education and how universities are handling this understanding. Such insight can also contribute to, and further spur, additional conversations and debate as to the direction of computer science education.

6 Acknowledgements

The author would like to thank the anonymous reviewers for their helpful comments, which were used to improve this article.

References

[1] ABET. Criteria for accrediting computing programs. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2021-2022/.

[2] ACM. An undergraduate program in computer science - preliminary recommendations. *CACM*, 8(2):543–552, 1965.

[3] ACM. Curriculum 68: Recommendations for Academic Programs in Computer Science. *CACM*, 11(3):152–197, 1968.

[4] ACM/IEEE-CS. Computer science curricular 2013. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[5] ACM/IEEE-CS/AAAI. Cs202x acm/ieee-cs/aaai comuter science curricula. https://csed.hosting.acm.org/.

[6] William F. Atchison and John W. Hamblen. Status of Computer Sciences Curricula in Colleges and Universities. *CACM*, 7(4):225–227, 1964.

[7] Richard Blumenthal. Using design patterns to address curricular growth issues. *Journal of Computing Sciences in Colleges*, 34(2):85–94, 2014.

[8] Glenn A. Bowen. Document Analysis as a Qualitative Research Method. *Qualitative Research Journal*, 9(2):27–40, 2009.

[9] Sebastian Dziallas and Sally Fincher. Acm curriculum reports: A pedagogic perspective. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, ICER '15, pages 81–89, New York, NY, USA, 2015. ACM.

[10] U.S. Department of Education International Affirs Office. Structure of the u.s. education system: Credit systems. https://www2.ed.gov/ous/usnei/us/credits.

[11] Thomas A. Keenan. Computers and Education. *CACM*, 7(4):205–209, 1964.

[12] T. Weldon. Reducing time to degree by cutting credit creep. https://knowledgecenter.csg.org/kc/system/files/cr_creditcreep.pdf.

Text Analysis of Research Topics From Nursing Journals*

Daniel McDonald[1], Karina McDonald[3] and Jace Johnson[2]
[1]Information Systems & Technology Department
[2]Woodbury School of Business
Utah Valley University
Orem, UT 84058
{daniel.mcdonald, 10697952}@uvu.edu
[3]Health Sciences Center School of Nursing
Texas Tech University
Lubbock, TX 79409
karmcdon@ttuhsc.edu

Abstract

Over the last five decades, with numerous advances in medicine and changes in patient payment models, the nursing profession has evolved. We analyzed 26,705 abstracts from three different nursing journals that spanned five decades, including the 1970s, the 1980s, the 1990s, the 2000s, and the 2010s. We extracted the research topics from the abstracts and aggregated them by decade. We compared research topics by decade and looked for dependencies between decades. We also looked for common research topics that spanned all five decades. With nursing research increasing in sophistication and quality, we looked for a greater focus on research within the journals. Finally, we looked for increasing coverage of nursing education in the research. We found that any decade of research topics was most like the following decade of topics. We also found a common core of topics that were shared across all five decades that included topics such as nurses, patients, diseases, injuries, and medical providers. Finally, we found a growing trend for research-related topics in the later decades and a declining trend for topics related to continuing education, education degrees, and education credit.

1 Introduction

In 2020, we celebrated the 200th year anniversary of the birth of Florence Nightingale. The Nursing profession has changed a lot over the last five decades. To work in hospitals, nurses are now required to get more education than ever. Where licensed practical nurses (LPN) or licensed vocational nurses (LVN) worked in the past, registered nurses (RNs) with a Bachelor of Science in Nursing degree are now required. More opportunities have opened for advanced practice registered nurses over the decades as well. In 1965, the first formal graduate certificate for nurse practitioners was created[9, 3]. In 1971, it was recommended that nurse practitioners be able to work as primary care providers[5]. In 2012, discussions between national certifying bodies, state boards of nursing, and accreditation agencies began to address the possibility of requiring a Doctor of Nursing Practice (DNP) degree as the minimum requirement for NP licensure.

We propose analyzing a sample of nursing research abstracts from the last five decades, including abstracts from the 1970s, 1980s, 1990s, 2000s, and 2010s. We will extract common nouns from the abstracts and combine them into categories or topics by decade. From there, we will analyze the changes in topics researched by decade. We will start with a brief literature review and then list our research questions. We will then discuss how our analysis was done in the methodology section. Finally, we will share the results of our analysis and make some conclusions.

2 Literature Review

Over the years, there have been several papers that looked at topics or trends in nursing research. Moody et al. analyzed nursing research specifically between the years 1977-1986. They found an increased use of sophisticated research methods and an increasing use of conceptual models[8]. In 2000, Choi, Song et al. examined research published in the Journal of Korean Academy of Nursing for 30 years. They analyzed many research-related areas including who conducted the research, who were the research subjects, what were the main themes, to what degree were empirical research methods used and how common was qualitative research. They found a definite methodological development over the time, but a lack of theory development. In 2020, Im, Sakashita et al. identified six themes in nursing research across five countries. The themes were demographic alterations, increasing diversities and globalization, technology innovation, individualized or personal care and population health initiatives and health policies and regulations, and nursing workforce changes[4].

Compared to what we reviewed in the literature, our research will span a

greater time frame and include more abstracts in the analysis. Different from other research, we will limit our analysis to research topics only and not look at methodologies or research subjects.

3 Research Questions

In looking at the research topics from the five decades, we propose the following four research questions:

1. Do topics trend over decades with adjacent decades having more topic similarities or are decades mostly topic independent?
2. What are the common topics across all decades?
3. Choi, Song et al. found an increase in the quality of nursing research over 30 years they studied[2]. Has there been an increase in research topics related to the research process?
4. As nurses have been required to get more education, has there been an increase in research topics related to nursing education?

4 Methodology

To conduct our topic research, we first identified nursing journals with a long publication history that continued to the present. Next, we accessed the PubMed abstracts for all the published abstracts from the selected journals. We then extracted all the topics from each research abstract and aggregate the topics by decade. Finally, we compared the topics between decades to answer our research questions.

4.1 Selecting Journals

There are some great resources for identifying nursing-related research. The Nursing and Allied Heath Resources Section (NAHRS) of the Medical Library Association created a selected list of nursing journals to assist librarians[10]. This list of 212 journals is extensive, but includes many specialized journals that are focused on specific topics such as plastic surgery, mental health, orthopedic nursing, pain management, vascular nursing, etc. Such focused research would make combining topics from the different journals less meaningful. The journals were also published from various countries, which would complicate our analysis. Finally, many of the journals were not in publication during the 1970s or 1980s, which makes the analysis of five decades impossible.

In 2006, Allen and Levy in "Mapping the general literature of American nursing" analyzed the core literature cited by "general" or popular United

States nursing journals. The three journals they used were American Journal of Nursing (AJN), Nursing, and RN[1]. In 1987, Skinner and Miller noted 1987 circulation numbers of 511,600 for Nursing, 330,428 for AJN, and 275,000 for RN. In their study, the three journals were the most commonly read journals by staff members at two different hospitals[11]. All three journals also had a substantial number of abstracts in the PubMed database. Of the nursing journals, AJN and Nursing had the fourth and fifth most abstracts in PubMed and RN had the sixteenth most out of 411 different nursing journals.

The three journals also had abstracts that mostly spanned the five decades. RN did not have anything for the decade of 2010. Table 1 shows the number of abstracts by decade for each of the three nursing journals. The journal RN has the least representation with 5,489 abstracts. The decade with the fewest abstracts overall is 1970-1979 with 4,447. The three journals together over the five decades have a total of 26,705 abstracts on PubMed.

Table 1: Nursing Abstracts by Decade and Journal

Decade	Nursing	RN	AJN	Total
2010-2019	1,920	0	2,803	4,723
2000-2009	2,304	1,149	2,457	5,910
1990-1999	2,662	1,470	2,104	6,236
1980-1989	1,852	1,582	1,955	5,389
1970-1979	857	1,288	2,302	4,447
Total	9,595	5,489	11,621	26,705

4.2 Downloading PubMed Abstracts

The authors downloaded 1,062 baseline files and 53 update files from the PubMed FTP site. The files were gzipped XML files. The authors wrote a program in Python to decompress the gzipped files, which were around 35 MB a piece, process the XML in the files, and insert the data into a SQL Server database.

Each compressed XML file contained around 30,000 PubMed citations. Instead of inserting all attributes of each citation into the database, the following nine attributes were extracted from each citation and added to a MS SQL Server table: ISOJournalAbbrev, abstractText, abstractTitle, authors, country, fileSource, journalTitle, language, and pubYear. With all the abstracts in MS SQL Server, the authors could use SQL to query specific journals over specific years.

4.3 Reading Abstracts and Extracting Topics

The authors used SQL to query the table of abstracts and extract the PMIDs that matched those from each of the five decades. The PMIDs were kept in five different text files. To process the abstracts for a particular decade, the text file was read, and the database was queried for each abstract in the list.

Each abstract was processed using our content tagging algorithm[7]. The tagging process uses an extensive dictionary of over two million word-tag entries. This tag dictionary has been augmented by previous research[6]. After tagging, the text is processed several times more to combine tags into topic categories. The topic categories are part of a large category hierarchy. There are over 5,000 different noun categories in the hierarchy into which terms (words or phrases) can be assigned.

For each decade, the authors extracted the noun topics from the abstracts. There were many different topics from each decade and the frequency distribution had a very long tail. To focus on the main topics for each decade, the authors focused just on the topics that accounted for 80 percent of the nouns from the abstracts. Table 2 shows the number of topics extracted from each decade and the number that was included in the analysis. For example, in the decade of 1970-1979, a total of 206 topics accounted for 80.47 percent of all the nouns. From that decade a total of 888 topics were extracted from the abstracts.

Table 2: Total Topics Extracted by Decade

Totals	1970s	1980s	1990s	2000s	2010s
Total topics extracted	888	914	950	1105	1184
Topics to analyze	206	212	227	264	229
Percent of total nouns	80.5%	80.2%	80.2%	80.2%	80.4%

4.4 Performing Topic Analysis

To answer our first research question about whether there is any topic dependency between decades or whether each topic is topically independent, we will utilize a variation of the information retrieval formula term frequency multiplied by inverse document frequency: tf x idf. For this calculation, we will use the categories that accounted for 80 percent of the nouns in the abstracts. Furthermore, we will not include the noun topics that are included in every decade, such as the NURSE topic and PATIENT topic. In total, there are 259 topics across the five decades excluding those that appear in every decade. If

two decades have an overlapping topic, then the term frequency will be 1 multiplied by the inverse of the total number of decades that contain that topic. For example, the topic of ROUNDS is only contained in the 1970s decade and the 1980s decade. The ROUNDS topic will contribute .5 to the similarity of the decades given the calculation of 1 x $\frac{1}{2}$. The document frequency is 2 because the ROUNDS topic only occurs in two different decades.

To address our second research question, we will look at the top 10 topics from every decade that are shared across all five decades within the top 50 percent of nouns. In other words, we want to focus on topics that are very common within a decade, but also very common across decades. For four decades the top 10 topics are shared across all the decades. For the decade of 2010, there are two topics in the top 10 that are not shared across all the decades, RESEARCH and LITERATURE.

For research question three, we will analyze the trends related to increasing focus on topics related to the research process across the five decades. The topic tags we will focus on include the following topics: RESEARCH, LITERATURE, QUESTIONNAIRE, INTERVIEW, AUTHOR, PUBLICATION, and DATA COLLECTION.

For research question four, we will analyze the research trends related to nursing education, both initial degrees and continuing education. The topic tags we will focus on include the following topics: EDUCATION, SCHOOL, CONTINUING EDUCATION, NURSING EDUCATION, EDUCATION DEGREE, and EDUCATION CREDIT.

5 Results

The results of our similarity analysis between decades of topics provided some evidence of dependence between decades. For example, for any decade, the most similar decade was the decade immediately following it. Table 3 shows

Table 3: Most Closely Matching Decades by Topics

Decade	Most Similar Decade	Similarity Score
1970s	1980s	14.53
1980s	1990s	17.03
1990s	2000s	20.70
2000s	2010s	24.12
2010s	2000s	24.12

the most similar decade to each of the five decades and their corresponding similarity scores.

Of the top five highest similarity scores, all were between adjacent decades except for one. The decade of 2000s had a similarity score of 15.37 with the 1980s decade. This high similarity score likely results in part because, as shown in Table 2, the decade had more topics in it than any other (a total of 264), at least 35 more topics than the next closest decade.

The top 10 topics from each decade that are shared across all five decades within the top 50 percent of all nouns are shown below, ordered by number of occurrences. There are 18 total topics that make up the top 10 topics from each decade. The top topic across 4 of the 5 decades is PATIENT, followed by NURSE, ROLE, DISEASE, AILMENT, and then NURSING.

- PATIENT: Patient(s), cancer patients, difficult patients, delicate patient, cardiac patient, elderly patient(s)
- NURSE: Nurse(s), RN, R.N., psychiatric nurse, clinical nurse, assertive nurse, registered nurses
- ROLE: Elderly, diabetic, student(s), specialist, assistants, graduates, consultant, advocate, expert, colleague, employee, victim, supervisor
- NURSING: Nursing, nursing care, nursing practice, ambulatory nursing, orthopedic nursing, primary nursing
- DISEASE & AIDS: Diabetes, leukemia, rheumatoid arthritis, scoliosis, Hodgkin's disease, cerebral palsy, Parkinson's disease, emphysema, cardiovascular disease, renal disease, COPD
- AILMENT: Stress, dysrhythmia, frostbite, hypertensive, decubitus ulcers, hyperalimentation, cardiac defects, hydrocephalus, psoriasis toxemia, glaucoma, dermatitis, kidney stones, diarrhea, diabetic coma, anemias, headache(s)
- PERSON: Clara Maass, Billy, A. Frank, Peter, Katie, Michael, Karen, Lisa, Jill, Jim, Janet, Willie, Harry, George, Brian
- SURGERY: Surgery, ostomy, cardiac surgery, total hip replacement, tracheostomy, open-heart surgery, mastectomy, vasectomy, craniotomy, coronary bypass, amputation
- TREAT: Drug therapy, treatment, hemodialysis, blood therapy, dialysis, immunotherapy, chemotherapy, radiation therapy, therapeutic touch, peritoneal dialysis
- CHILD: Child, children, kids, troubled child, foster child, hyperactive child
- PROBLEM: Problem(s), crisis, trouble, pitfalls, dilemma(s)
- CARE: Care, burn care, respite care, trach care, hospice care, cataract care, physical care, post-op care, private care, self-care, preventive
- INJURY: Burn(s), injuries, pressure sores, wound(s), trauma, spinal cord injury, gunshot wounds, draining wounds
- INSTRUCT: Instruction, teaching, tip(s), lesson(s), lectures, order(s)

- MEDROLE: Physician, practitioner, pharmacist, caregiver(s), nutritionist, doctor(s), clinician, surgeon, therapist, psychiatrist, midwife, epidemiologist, pediatrician(s), anesthesiologist
- MEDICINE: Pill, medicine, geriatric medications, penicillin, nitroglycerin, prescription, ketamine, Levodopa, Rx
- PAIN: Pain, suffering, intractable pain, low back pain, chronic pain, leg pain severe pain, chest pain, menstrual discomfort, lumbar aches
- GROUPING: Series, group(s), focus group(s), support group(s), state nursing groups, activity groups

From the above list, there is insight into what nurses do based on the top topics in the research. Nurses help ailing patients that are in pain with ailments, diseases, and injuries. Nurses provide all kinds of care and treatment to the patients and do so in cooperation with other medical professionals such as doctors, pharmacists, and other practitioners. In addition, nurses administer all kinds of medicine and give instructions to patients in how to care for their medical conditions.

To address the third research question, the authors looked at the following seven topics: RESEARCH, LITERATURE, QUESTIONNAIRE, INTERVIEW, AUTHOR, PUBLICATION, and DATA COLLECTION. Table 4 shows whether the topics were within the top 80 percent of noun occurrences in the five decades.

Table 4: Research-Related Topics Over the Decades

Topic	1970s	1980s	1990s	2000s	2010s
RESEARCH	1	1	1	1	1
LITERATURE	0	0	0	1	1
QUESTIONNAIRE	1	0	1	1	1
INTERVIEW	0	0	1	1	1
AUTHOR	0	0	0	1	1
PUBLICATION	0	0	0	1	1
DATA COLLECTION	0	0	0	0	1

While the topic of RESEARCH is covered in all five decades, the prevalence of other research-related topics such as LITERATURE, INTERVIEW, AUTHOR, and PUBLICATION really don't show up until the later decades. There is some evidence that there is an increasing focus on research in the nursing journals we analyzed.

To address the fourth research question, the authors looked at the following six topics: EDUCATION, SCHOOL, CONTINUING EDUCATION, NURSING EDUCATION, EDUCATION DEGREE, and EDUCATION CREDIT.

Table 5 shows whether the topics were within the top 80 percent of noun occurrences in the five decades. The results are a bit mixed in the education-related topic area. While the general topic of education is consistent across the decades, it does include things like "sex education" and "diabetes education". On the other hand, specific references to continuing education, education degree and education credit seem to be decreasing. Perhaps with the educational standards established early, the need to continually address the topics decreased. There is some evidence that the focus on nursing degrees and school credit have gone down over the decades in the journals we analyzed.

Table 5: Education-Related Topics Over the Decades

Topic	1970s	1980s	1990s	2000s	2010s
EDUCATION	1	0	1	1	1
SCHOOL	1	0	0	1	1
CONTINUING EDUCATION	1	1	0	0	0
NURSING EDUCATION	1	0	0	0	1
EDUCATION DEGREE	1	1	0	0	0
EDUCATION CREDIT	0	1	0	0	0

6 Conclusions

We analyzed 26,705 abstracts from three different American-based nursing journals that spanned five decades. We found that topics from a decade were most like the topics from the decade that followed. We also found a strong core of common nursing topics across all five decades that depicted well the concern of nurses. We found some evidence that the focus on research topics is increasing over the decades and the focus on educational degrees and continuing education in research topics is decreasing.

In future research, we want to analyze how topics are related through co-occurrence algorithms and display the relationships in a graph to visualize the relationships.

References

[1] M.P. Allen and J.R. Levy. Mapping the general literature of american nursing. *Journal of the Medical Library Association*, 94(2 Suppl):E43–48, 2006.

[2] K. Choi, M.S. Song, A.R. Hwang, K.H. Kim, M.S. Chung, S.R. Shin, and N.C. Kim. The trends of nursing research in the journal of the korean academy of nursing. *Journal of the Korean Academy of Nursing*, 30(5):1207–1218, 2000.

[3] L. Ford. A nurse for all settings: The nurse practitioner. *Nursing Outlook*, 27(8):516–521, 1979.

[4] E.O. Im, R. Sakashita, C.C. Lin, T.H. Lee, H.M. Tsai, and J. Inouye. Current trends in nursing research across five locations: The united states, south korea, taiwan, japan, and hong kong. *Journal of Nursing Scholarship*, 52(6):671–679, 2020.

[5] A.W. Keeling. Historical perspectives on an expanded role for nursing. *The Online Journal of Issues in Nursing*, 20(2), 2015.

[6] D. McDonald. A system for automatic lexical acquisition from pubmed. *Journal of Computing Sciences in Colleges*, 36(2):71–78, 2020.

[7] D. McDonald, H. Chen, H. Su, and B. Marshall. Extracting gene pathway relations using a hybrid grammar: the arizona relation parser. *Bioinformatics*, 20(18):3370–8, 2004.

[8] L. Moody, M.E. Wilson, K. Smyth, R. Schwartz, M. Tittle, and M.L. Van Cott. Analysis of a decade of nursing practice research: 1977-1986. *Nursing Research*, 37(6):374–379, 1988.

[9] National Women's Hall of Fame. Loretta C. Ford. `https://www.womenofthehall.org/inductee/loretta-c-ford/`.

[10] P. Sherwill-Navarro, J.C. Kennedy, and M.P. Allen. Developing an evidence-based list of journals for nursing. *Journal of the Medical Library Association*, 102(2):105–109, 2014.

[11] K. Skinner and B. Miller. Journal reading habits of registered nurses. *The Journal of Continuing Education in Nursing*, 20(4):170–173, 1989.

Knowledge Sharing Technology in School Counseling: A Literature Review*

Kalee Crandall
Dakota State University
Madison, SD 57042
kalee.crandall@trojans.dsu.edu

Abstract

School counselors are expected to perform a wide range of tasks to improve student outcomes but are oftentimes limited in the resources needed to perform these tasks. The lack of resources, including time and the knowledge needed to complete tasks effectively, may contribute to unnecessary stress and possible burnout. This research uses a modified systematic literature review process to explore knowledge sharing technology in school counseling and presents a proposed model for future research adapted from Alavi's Model of Knowledge Transfer among Individuals in a Group. The findings of this study outline the current state of research and are relevant to research and practice.

1 Introduction

The roles and responsibilities of the modern school counselor are vast and extensive. The services they provide range from ensuring students are college and career ready to support their social and emotional well-being. In addition, school counselors use data to promote equity in the services they offer students and strive to improve learning for every student in the school population[23].

Given the quantity and significance of these responsibilities, it is not surprising that many school counselors experience stress, which can lead to job dissatisfaction and burnout[11]. Many school counselors who choose to not return to their jobs report not only more stress but also report a lack of resources[10].

Because it is unlikely that the demands of school counselors will decrease, McCarthy[10] proposes that increasing resources may reduce the stress of school counselors.

According to the Knowledge-Based Perspective of the Firm Theory, knowledge is the most strategically significant resource of the firm. Knowledge is information possessed in the mind of individuals. It is personalized information related to facts, procedures, concepts, interpretations, ideas, observations, and judgments[2]. Knowledge sharing occurs when individuals are willing and able to collaborate with others[6].

Although educators are seeking to understand how they can be more effective in collecting, disseminating, sharing information, and seeking better ways to transform knowledge into effective decision-making[17], little research has been done on knowledge sharing in education, specifically in school counseling. This literature review will explore this topic and answer the call from information systems academics for further exploration and the barriers impeding knowledge sharing[6].

2 Methods

In this paper, a simplified literature review process has been adapted from Okoli[15] to help extract literature from these databases so that only the most relevant literature is used and to ensure all identified published research is relevant to the research question, "How can knowledge sharing in school counseling be improved using technology?"

To help identify previous research on this topic the terms, "knowledge sharing" with "school" "counseling" were used to find published peer-reviewed articles within the last five years in the following databases: Google Scholar, ProQuest, and ACM Digital Library. There were 110 articles identified in the search process.

The protocol for the search is to identify published journal articles relating to the research question using the above-mentioned key terms, as shown in Figure 1. Publications that did not include these terms in the title, abstract or subject/key terms were excluded from the results. Additionally, duplicate articles were also excluded from the results. Eight published papers remained for the final analysis of this literature review.

3 Results

Conducting a full textual analysis assists in appraising the quality of the publications as well as synthesizing the studies as proposed by Okoli[15]. The remaining articles revealed the recent research areas of knowledge sharing practices among school counselors. Simons[21] discussed theories and knowledge sharing practices among school counselors drawing upon previous research[20]

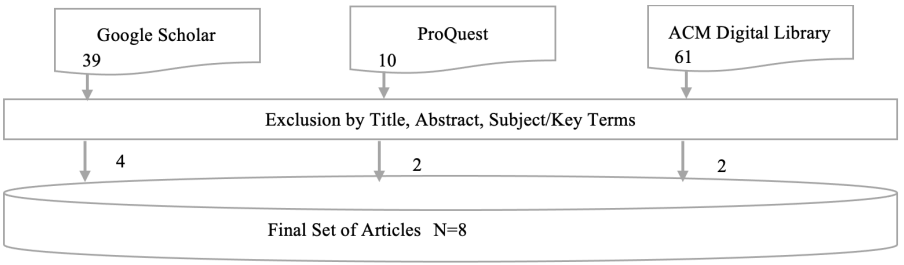


Figure 1: Extraction process to identify relevant publications

also examined this topic. Wilkes, Nobe, Clevenger, and Cross[10] described how knowledge gaps impact counselors' ability to offer guidance related to certain careers. These researchers propose a program to specifically address this issue, with the first phase of the program focusing on knowledge sharing.

Social processes and communities have also been extensively researched. Brayton[5] discussed perceptions of knowledge sharing in higher education, finding social processes facilitated significant engagement in knowledge sharing activities, and subsequent knowledge development. Additionally, Phusavat, Delahunty, Kess, & Kropsu-Vehkapera[18] explained how educators become more motivated by participating in communities that support knowledge sharing.

Technology facilitating knowledge sharing practices has also been studied and developed to support counselors in their responsibilities. StressMon[27] is a stress and depression detection system that leverages single-attribution location data sensed from the WiFi infrastructure. This technology can assist counselors by improving awareness of mental health issues and provide better workflow management. One of the major features of this system is the total time spent on an activity and the number of times a user engaged in that activity per day. Knowledge sharing is listed as a domain-specific workgroup activity. LEAP[25] facilitates conversation and the exchange of knowledge around a given topic with access to learning material. Users of this system exhibit benefits of collaborative learning in terms of positive interdependence on each other and the use of interpersonal skills.

Two other articles were identified in the search process, and while the content did not directly relate to school counselors, the results of the research could indirectly inform future research. Shepherd and Cooper[19] explored the factors influencing knowledge creation and knowledge sharing in a high velocity, networked environment. Although the environment in this study was not school counseling, the results may still pertain to school counseling, where

counselors must be able to develop constructive and cooperative working relationships with others and maintain them over time[14]. Muro, Stickley, Muro, Blanco, and Tsai[12] offered clarification and a description of simple shared resources by providing shared personal experiences and exploring relevant issues relating to the collaboration between schools. While this article also did not directly relate to school counseling, the sharing of resources and collaboration is essential to knowledge sharing.

4 Theoretical Development

It is clear from the review of the articles that more research is needed, specifically to understand the role technology plays in knowledge sharing practices amongst school counselors to reduce their stress. To guide future research endeavors, it is essential to have an understanding of pertinent theories relating to knowledge sharing and the unique characteristics of the target population.

According to Lazarus and Folkman[8], when life demands are encountered, a cognitive balancing act ensues, in which perceived demands are weighed against the resources one has for coping with demands. In other words, stress results from an imbalance of perceived demands and resources available. One of the most abundant and important resources a school counselor has is knowledge.

The knowledge-based theory of the firm is an extension of the resource-based theory of the firm[16] and focuses on problem-solving and knowledge formation by continually discovering new knowledge or new solutions that come from combinations of existing knowledge[13]. In this theory, managers choose problems while identifying knowledge sets or existing technology that are potentially useful in searching for solutions to that problem. This theory has been useful in explaining how organizations can gain a competitive advantage by effectively applying the existing knowledge to create new knowledge.

Creating this new knowledge requires knowledge transfer. Alvani[2] proposed a framework to explain knowledge transfer. In this model, arrows represent the process of knowledge application, learning, or new knowledge creation that occurs when individuals apply knowledge and observe the results. Transfer of knowledge occurs at various levels, between individuals, from individuals to explicit sources, from individuals and groups, across groups, and from the group to the organization. Once individual A transfers knowledge to individual B, individuals B's knowledge processes may be triggered, leading to knowledge creation. Individual B can then chose to apply the knowledge, consult with other members, or record the knowledge.

Even though studies have shown the importance of knowledge sharing, oftentimes, knowledge sharing processes do not occur, and there is a lack of research exploring these barriers[6]. Studies have shown, however, there is a significant and positive association between knowledge sharing processes and

information technology[1]. Azari and colleagues[3], found technology was one of the independent predictors of knowledge sharing behavior, which may prove problematic for school counselors.

While technology can assist in knowledge sharing practices among school counselors, many are cautious of incorporating new technology within their profession[23]. One possible explanation for this resistance is the lack of comfort and skill school counselors have learning and using technology[26, 22], resulting in low levels of self-efficacy.

Self-efficacy[4] is a person's belief in their ability to succeed in a particular situation. Unless school counselors believe they can produce desired outcomes by their actions, they have little incentive to act, or in this case, use technology.

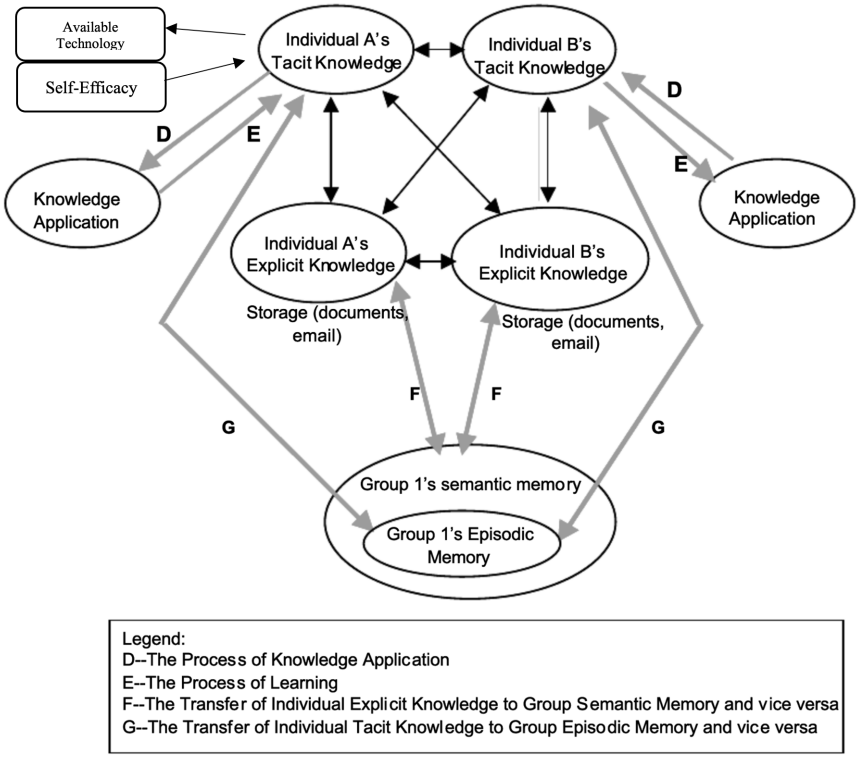


Figure 2: Proposed model for future research adapted from Alavi (2001)

The model proposed in Figure 2 extends Alavi's[2] framework by adding two additional constructs, self-efficacy, and available technology, to increase knowl-

edge transfer among school counselors. Although knowledge sharing may improve school counselor's stress levels, they may not participate in this beneficial practice if the technology is not available or self-efficacy using the technology is low. This model can guide future researchers in understanding knowledge sharing practices in school counseling.

5 Evaluation

To evaluate the proposed model, researchers need to understand the role and availability of technology to facilitate the knowledge sharing process in school counseling, as well as counselors acceptance of the technology. While research alludes to the idea that counselors have low self-efficacy when it comes to technology, more can be done to understand why that is still the case and if it impedes the knowledge sharing process. Future research questions may include the following:

- How are counselors currently sharing knowledge?
- How can knowledge sharing reduce the stress of school counselors?
- What are the design principles needed for technology acceptance of school counselors?
- What technology is available for school counselors to share knowledge?

6 Discussion & Conclusions

Knowledge is one of the most important resources a school counselor has to deal with stress effectively. According to Pfeffers [7], individual success in organizations is quite frequently a matter of working with and through other people, and organizational success is often a function of how successfully individuals can coordinate their activities.

However, despite the benefits of knowledge sharing, it is rarely implemented in school counseling. This paper explores recent studies relating to knowledge sharing in school counseling. Despite the amount of research, further studies are needed to improve the knowledge sharing processes in this unique population. In this paper, a review of previous literature was presented with areas for future research identified, specifically research focusing on technology acceptance and availability. Using the extension of Alavi's [2] Model of Knowledge Transfer among Individuals in a Group presented in this paper can guide future researchers in this endeavor and advance the discussion in the information systems community about the next generation of knowledge sharing technologies.

References

[1] K. K. Al-Boquor. The role of information technology and knowledge sharing and their impact on higher education quality assurance 'an empirical study at Taif University'. Dirasat: Administrative Sciences, 43(1), 2016.

[2] M. Alavi and D. E. Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly*, 25(1):107-136, 2001, doi: 10.2307/3250961.

[3] G. Azari, L. Riahi, and R. Dehnavieh. The status of knowledge sharing at the decision-making center of deputy of education in Iran's ministry of health and medical education. *Iranian Journal of Medical Education*, 16, 2016.

[4] A. Bandura. *Social foundations of thought and action: A social cognitive theory.* Englewood Cliffs, NJ: Prentice Hall, 1986.

[5] S. W. Brayton. *Participant perceptions of knowledge sharing in a higher education community of practice.* ProQuest Dissertations Publishing, 2016.

[6] Y. Charband and N. J. Navimipour. Erratum to: Online knowledge sharing mechanisms: A systematic review of the state of the art literature and recommendations for future research. *Information Systems Frontiers*, 21(4):957-957, 2019/08/01 2019, doi: 10.1007/s10796-017-9799-2.

[7] P. Jeffrey. Understanding power in organizations. *California Management Review*, 34(2):29-50, 1992, doi: 10.2307/41166692.

[8] R. S. Lazarus and S. Folkman. *Stress, appraisal, and coping* New York: Springer, 1984.

[9] E. Mason, C. Griffith, and C. Belser. School counselors' use of technology for program management. *Professional School Counseling*, 22(1), 2018, doi: 10.1177/2156759X19870794.

[10] C. McCarthy, V. Van Horn Kerne, N. A. Calfa, R. G. Lambert, and M. Guzmán. An exploration of school counselors' demands and resources: Relationship to stress, biographic, and caseload characteristics. *Professional School Counseling*, 13(3):146-158, 2010 `http://www.jstor.org/stable/42732888`

[11] P. R. Mullen, A. J. Blount, G. W. Lambie, and N. Chae. School counselors' perceived stress, burnout, and job satisfaction (German). *Professional School Counseling*, 21(1), Sep 2017-Aug 2018, doi: `http://dx.doi.org/10.1177/2156759X18782468`

[12] J. H. Muro, V. K. Stickley, L. Muro, P. J. Blanco, and M.-H. Tsai. Considerations for elementary schools and university collaborations: a guide to implementing counseling research. *The Journal of Research Administration*, 46(1):102, 2015.

[13] J. A. Nickerson and T. R. Zenger. A Knowledge-Based Theory of the Firm—The Problem-Solving Perspective. *Organization Science*, 15(6):617-632, 2004, doi: 10.1287/orsc.1040.0093.

[14] O*NET. Details Report for: 21-1012.00 - Educational, Guidance, School, and Vocational Counselors. `https://www.onetonline.org/link/summary/21-1012.00`

[15] C. Okoli. A guide to conducting a standalone systematic literature review. *Communications of the Association for Information Systems*, 37, 2015, doi: 10.17705/1CAIS.03743.

[16] E. T. Penrose. *The Theory of the Growth of the Firm* Oxford, UK: Oxford University Press, 1959.

[17] L. Petrides and T. Nodine. Knowledge Management in Education: Defining the Landscape. Institute For the Study of Knowledge Management in Education, 2003.

[18] K. P. Phusavat, D. Delahunty, P. Kess, and H. Kropsu-Vehkapera. Professional/Peer-learning community. *Journal of Workplace Learning*, 29(6):406-427, 2017, doi: 10.1108/JWL-11-2016-0098.

[19] A. Shepherd and J. Cooper. *Knowledge Management for Virtual Teams*. IACIS Europe, Warsaw, Poland, 2020.

[20] A. Shipp. *An exploration of school counselors' knowledge sharing practices using diffusion of innovation theory, social exchange theory, and theory of reasoned action (Doctoral disseration)*. ProQuest, UMI Dissertations Publishing, 2010.

[21] J. D. Simons, B. Hutchison, and M. W. Bahr. School counselor advocacy for lesbian, gay, and bisexual students: Intentions and practice. *Professional School Counseling*, 20(1a):29-37, 2016, doi: 10.5330/1096-2409-20.1a.29.

[22] T. M. Steele, D. E. Jacokes, and C. B. Stone. An examination of the role of online technology in school counseling. *Professional School Counseling*, 18(1):125-135, 2015

[23] Utah State Board of Education. *College and Career Readiness School Counseling Program Model*. Addison-Wesley, 2nd edition, page 80, 2016. Available: `https://www.schools.utah.gov`

[24] J. Wilkes, M. C. Nobe, C. Clevenger, and J. Cross. Needs assessment: Identifying and addressing high school counselors' perceptions of construction management. *International Journal of Construction Education and Research*, 11(3):196- 217, 2015.

[25] D. Yadav, A. Bhandari, and P. Singh. LEAP: Scaffolding collaborative learning of community health workers in India. In *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW, 169):1-27, 2019.

[26] A. Young and C. Kaffenberger. School Counseling Professional Development: Assessing the use of Data to Inform School Counseling Services. *Professional School Counseling*, 19(1):46-56, 2015.

[27] C. Zakaria, R. Balan, and Y. Lee. StressMon: Scalable detection of perceived stress and depression using passive sensing of changes in work routines and group interactions. In *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW, 37):1-27, 2019.

Instruction Delivery Modes and Learning Experiences in COVID-19 Pandemic*

Ajay Bandi
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
ajay@nwmissouri.edu

Abstract

In response to COVID-19 pandemic measures, universities mitigated the instruction delivery mode overnight. The transformation brings several difficulties for faculty to teach and students to learn. This research aims to present the different mitigation strategies in instruction delivery and identify the benefits and challenges of student learning in the pandemic. The traditional instruction model is unsuitable in the pandemic. A-Zoom-enabled virtual classrooms are suitable as most students are willing to attend classes from home for their safety. A survey data with an 88.8% response rate is synthesized and categorized various benefits and challenges of remote learning. Educators benefit from the results on how to design and deliver their classes in the pandemic.

1 Introduction

Despite the advances in the technology and medical fields, human beings are still vulnerable to an outbreak of novel pathogens. One such example is the recent outburst of the coronavirus (SARS-CoV-2) that affected many people worldwide. This health emergency crisis hit the educational institutions at all levels, from pre-school to the post-graduation. The administrations offer a wide variety of instruction modes, including the face-to-face and hybrid instruction delivery methods [7]. At a university level, the instruction mode

selection depends on various factors such as class size, student classification (freshmen, sophomore, junior, senior, graduate), and the nature of the class (theory, hands-on programming, laboratory setting, tool-based). Instructors must also accommodate students' education needs in quarantine and isolation due to infectious diseases' vulnerability. Therefore, the transition is not easy [13] and has several challenges for students to learn and educators to teach and maintain student engagement [5, 15]. Also, there is no single mitigation strategy that operates for the entire education system. It entirely depends on the number of cases reported in that location and the local governments' rules.

In the literature, researchers focused on general online learning and teaching higher education issues [9, 11, 14, 18] and their impacts [6] on students. However, there is a need to concentrate on the various modes of delivering instruction to computer science students other than existing techniques. This paper's contribution is two-fold in reporting the survey results. (i) Students' experiences in the pandemic, who enrolled in hands-on programming and tool-based courses. (ii) Instructors' experiences designing, developing, and teaching those courses in the pandemic.

2 Related Work

Researchers presented their experiences and lessons learned in their online pedagogy of cybersecurity [1] introductory Computer Science [16], and software engineering [10] courses when compared to traditional classroom pedagogy [2, 3].

Bai, Gao, and Goda [1] lessons learned from teaching cybersecurity classes, pointed out that the students cannot perform their internships and are replaced that requirement with 'jobs related skills' course as a temporary adjustment. Also, challenges were raised in teaching networking labs, and they were conducted in a segregated mode. Chhetri [4] in the study of community college students, presented that students showed interest in remote learning with benefits such as video conferencing and class recordings. At the same time, they experience challenges with self-learning and staying on track of due dates. However, assessments are another challenge for instructors to evaluate students' performance fairly. In their survey study, Motogna, Marcus, and Molnar [10] administered the questionnaire to instructors worldwide. Their results showed that the instructors' efforts are increased due to the transformation of online learning. They also observed that there is a trend in reducing the exams and growth in project-based evaluations. Seeling [16], in their introductory programming classes, experienced that students' participation has reduced in completing the ZyBook examples for more difficult concepts.

Digital multimedia technology tools [12, 14] for video conferencing (Zoom, Microsoft Teams, Google Meet, Cisco Webex) [18] and learning management

systems (Canvas) [8] plays a significant role in this COVID-19 pandemic. The transition from traditional education experiences to virtual environment [17] has psychological [6] impact on the students. However, these studies lack the specifics of the modes of instruction delivery methods remote teaching and learning. This research address the modes of instruction delivery methods remote teaching and learning in computer science courses and present the benefits and challenges of students' experiences.

3 Modes of Remote Instruction

At the beginning of the pandemic in March 2020, the university administration provided options to adapt one of these instruction delivery modes to the faculty as COVID-19 teaching mitigation strategies. The same methods are used in fall 2020. It is up to the instructors to choose the appropriate mitigation technique for their classes. Overall, in all the mitigation strategies of instruction delivery, students' attendance policy is relaxed without class participation points, such as surprise quizzes, bonus points for attending classes, etc.

- Alternating attendance – synchronous: Alternating attendance in which students attend one day in-person and participate in the course remotely using Zoom (or other technology) at the class time.
- Alternating attendance – asynchronous: Alternating attendance in which students attend one day and participate in the course remotely using Zoom (or other technology) within a time frame established by the instructor (asynchronous).
- Split sections: Divide one larger section into two. For example, split sections would be taking one section of 50 students and changing it into two sections of 25 students each. There would be no load change for faculty.
- Multiple locations: Multiple locations would be taking a section of 50 students and having 25 students meet in one room and the other 25 in another room, but at the same time. The faculty member can only be in one place at a time and might Zoom to the other room or go back and forth. Again, no-load change occurs.
- No change: Face-to-face delivery each day with face covering and maintain six feet social distancing. "No change" often meant a lenient attendance policy and enabled Zoom during the class session. This effectively looked a lot like alternating attendance – synchronous, but without structured alternating attendance.

- Online: The online-only classes that were taught before the pandemic. For other courses and zero credit hour labs to teach online, Provost's approval is necessary.

4 Research Method

The author administered a concise survey in two different classes — object-oriented programming, and data visualization. The first course is hands-on programming requires demonstration and execution of source code. Data visualization is a tool based course also requires demonstration for connecting data sets, visualizing appropriate charts, and exporting the results.

Google forms and spreadsheets are used as survey instrument tools. The survey is not part of any assignment, and below are the two simple survey questions given to students. To avoid bias, the authors gave this survey after students complete the instructor course evaluations. The response rate of the survey is 88.8% (40 responded out of 45 students).

1. What are things that worked out well in the learning and teaching materials in the pandemic?

2. What are the things that can be improved in teaching the course in the pandemic?

5 Results and Discussion

At the beginning of the fall semester, the authors adapted alternating attendance – synchronous mitigation strategy. The students attended alternate days for the first couple of weeks, and then, due to quarantine and isolation, students started connecting through Zoom. Whenever students attend classes in-person, the faculty member recorded the attendance and the student's location in the classroom for future contact tracing. As the semester moving forward, more students are attending through Zoom. The following are the changes for remote teaching compared to the traditional classroom pedagogy.

- Learning outcomes: The learning outcomes of the courses are not modified. The author did not alter or compromise the outcomes related to communication and teamwork. However, there is an obvious modification in assessing students.

- Assessments: All the replaced in-person classroom closed book exams with take-home exams and increasing the number of assignments and projects. As a result, the grading time has also increased.

- Student engagement: The goal is that students gain a similar interactive experience of in-person classroom participation. To achieve this, we incorporated threaded discussions, breakout sessions, and chats in Zoom. Padlet is used to organize student's interactions in affinity diagrams. We also used in-class responsive software such as Poll Everywhere and Mentimeter to grab students' attention in remote teaching.

- Team Projects: The author helped divide tasks individually for each team member and guided teams to finish their projects successfully. Finally, students recorded their presentations. Mentoring course projects remotely takes a significant amount of time, which increases the faculty workload.

- Delivering lectures: The author delivered lectures through Zoom in the classroom with students who wanted to attend in-person classes. The tool demonstrations and program executions were prerecorded and uploaded to YouTube for student's reference.

Table 1: Benefits and Challenges of Remote Learning

Benefits	Challenges
• Online YouTube videos • Virtual Meetings • Motivation • Collaboration • Gaining new skills (writing and multimedia)	• Lack of resources • Assessments • Student engagement • Keeping track of schedules • Lack of introducing new topics

The survey data is synthesized using coding techniques in ground theory [19] and presented the benefits and challenges of remote learning in a tabular format in Table 1.

5.1 Benefits of Remote Learning

This subsection presents the instructor's reflection on each benefit and a few selected responses from each category's survey data. The rationale for giving these categories is that more than 50% of the students recognized these benefits.

1. *Online YouTube videos:* The prerecording of the demonstrations of various types of charts in Tableau/Excel[1] and the executions of core Java[2]

[1] https://www.youtube.com/playlist?list=PLu3nilPxiKYEQLj2EmRciqX8KH095HOox
[2] https://www.youtube.com/playlist?list=PLu3nilPxiKYE5xT6OIuWYmpC9fQVE2ARO

and Java Spring Boot[3] programs helped students finish their assignments. Those videos are uploaded to YouTube channel. Students can view them at any time.

"The instructional YouTube helped a lot because they walked me through each demo." "The video examples/tutorials went well. Mainly because I'm more of a visual learner, so that helps a lot." "Watching the videos and following along in Excel/Tableau."

2. *Virtual Meetings:* Students recognized the importance of virtual Zoom meetings and appreciated for conducting them.

   "Virtual meetings and recorded sessions were good."

3. *Motivation:* Interestingly, at least three student teams self-motivated for team projects and completed their tasks in visualizing real-time data or streaming data. Streaming data is the current topic and not covered in the class in detail. The instructor role is to mentor the teams and assign specific tasks to students.

   "The best is the Data Visualization project. Myself, I have Improved, implemented, and learned new skills(Kafka). Searched and explored new possible solutions to complete the tasks."

4. *Collaboration:* Virtual Zoom enabled sessions helped students to collaborate and finish their projects. Students observe that virtual team meetings encouraged them to meet with team members and work together by sharing screens.

   "Working as a team for a project without even meeting teammates directly which was thought impossible became possible"

5. *Gaining new skills:* The learning outcomes related to communication is not compromised in remote learning. Students are required to record their presentation work and upload it to Canvas or YouTube. Students recognize the importance of making and editing videos. Also, for learning from the home environment, writing skills are essential to communicate with others.

   "I have learned a lot in this course. Even with few online meetings and new learning methodology, the course was effective, and I have obtained good amount of knowledge through this class. I have also learned to communicate through mails which is very important in my future."

   "We have improved a lot in writing emails as most of the communication went through emails. And the other thing that worked out well in this

[3] https://www.youtube.com/playlist?list=PLu3nilPxiKYE1W2-8jrP8_AAapRIrPub6

pandemic is digitizing things. We familiarized with many tools related to attend the meetings, etc."

5.2 Challenges of Remote Learning

This subsection presents the instructor's reflection on each challenge and a few selected responses from each category's survey data. The rationale for giving these categories is that more than 50% of the students recognized these challenges.

1. *Lack of resources:* A few students found it time-consuming to watch and perform the executions as they do not have extra monitors other than laptops. Some students face internet connection issues, as well.

   "It is hard to watch videos and work simultaneously as we don't have dual monitors."

2. *Assessments:* Due to the pandemic, we have replaced closed book exams with increased assignments and take-home exams. A few students do not want to do more projects; instead, they suggest having multiple-choice tests.

   "Different variety of assignments, the Excel and Tableau assignments were fine, but I think maybe a couple more discussion assignments or multiple choice assignments would help students understand the material better."

3. *Student engagement:* To mimic in-person classroom engagement in remote learning, instructors added threaded discussion assignments. Students faced issues that a few students are not contributing significantly.

   "Felt like it was a pretty straight forward class, but I guess discussions were kind of awkward in my opinion. Mainly because once someone already has answer the question with the exactly same answer that you were going to say, it becomes either an I agree or finding some other way to answer the discussion situation."

   "More in person Zoom classes to promote student engagement."

4. *Keeping track of schedules:* It is a common challenge that students face in traditional face-to-face classes. Students faced problems keeping track of deadlines and due dates in the remote class setting. A majority of students want to have flexible deadlines for their assignments.

   "It would be helpful to know of all assignments ahead of time. It is hard to keep track of the schedules and deadlines."

5. *Lack of introducing new topics:* Students expect the currently trending topics or tools to be taught in classes. However, due to remote instruction, lecture materials for such topics are limited.

   "More demo examples on sorting and collections frameworks"

   "Some advanced techniques need to cover as part of the curriculum such as drill-down and drill-up techniques etc."

   "Introduction to more tools other than PowerBI and Tableau"

6 Limitations

Students' involvement in remote learning depends on various factors such as previous learning experience, specific future anticipations from the course outcomes, course design, delivery method, and availability of resources (textbooks, hardware, software, etc.). A large extent of the survey to be conducted for statistically significant evidence. However, this research presents the preliminary results of the impact of remote teaching and learning.

7 Conclusion

The benefits and challenges from the students' learning experiences will help the technology educators teach in unprecedented situations. The traditional model of instruction and the mitigation techniques presented in this article are not suitable. The alternating attendance synchronous style turned into Zoom enabled virtual classroom as most students were willing to attend classes from home due to COVID-19. The survey data shows that uploading videos of the source code executions and the concept demos into YouTube helped students watch them at any time to finish their assignments. Students encountered challenges of not having flexible due dates, extra assignments & projects, and longer take-home exams. Future work involves conducting a comprehensive survey and reviewing the student's and faculty's insights in learning and teaching computer science and information technology courses in the pandemic.

References

[1] Yan Bai, Chunming Gao, and Bryan Goda. Lessons learned from teaching cybersecurity courses during covid-19. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 308–313, 2020.

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

[3] Ajay Bandi, Abdelaziz Fellah, and Harish Bondalapati. Embedding security concepts in introductory programming courses. *Journal of Computing Sciences in Colleges*, 34(4):78–89, 2019.

[4] Chola Chhetri. "I Lost Track of Things" Student Experiences of Remote Learning in the Covid-19 Pandemic. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 314–319, 2020.

[5] Tom Crick, Cathryn Knight, Richard Watermeyer, and Janet Goodall. The impact of covid-19 and "emergency remote teaching" on the uk computer science education community. In *United Kingdom & Ireland Computing Education Research conference.*, pages 31–37, 2020.

[6] Roberta De Michele. Benefits, drawbacks and psychological impact of online lectures during quarantine due to covid-19 pandemic. In *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*, pages 257–260, 2020.

[7] Hollis Greenberg, Dan Bogaard, C Derrick Huang, Tim Preuss, and Cara Tang. Panel: What covid-19 is teaching professors: Pandemic-level changes in our classrooms. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 291–292, 2020.

[8] Matthew Seth Helmandollar. Meeting students where they are: Implementing canvas for successful student outreach. *Inquiry*, 23(1):n1, 2020.

[9] Svetlana Kurbakova, Zlata Volkova, and Alexander Kurbakov. Virtual learning and educational environment: New opportunities and challenges under the covid-19 pandemic. In *2020 The 4th International Conference on Education and Multimedia Technology*, pages 167–171, 2020.

[10] Simona Motogna, Andrian Marcus, and Arthur-Jozsef Molnar. Adapting to online teaching in software engineering courses. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*, pages 1–6, 2020.

[11] Emily Nordmann, Chiara Horlin, Jacqui Hutchison, Jo-Anne Murray, Louise Robson, Michael K Seery, and Jill RD MacKay. Ten simple rules for supporting a temporary online pivot in higher education, 2020.

[12] Debajyoti Pal, Vajirasak Vanijja, and Syamal Patra. Online learning during covid-19: Students' perception of multimedia quality. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–6, 2020.

[13] Sudhaman Parthasarathy and San Murugesan. Overnight transformation to online education due to the covid-19 pandemic: Lessons learned. *eLearn*, 2020(9), 2020.

[14] Christina Romero-Ivanova, Michael Shaughnessy, Laura Otto, Emily Taylor, and Emma Watson. Digital practices & applications in a covid-19 culture. *Higher Education Studies*, 10(3):80–87, 2020.

[15] Steven Schmidt, Elizabeth M Hodge, and Christina M Tschida. How instructors learn to teach online: Considering the past to plan for the future. *eLearn*, 2020(10), 2020.

[16] Patrick Seeling. Switching to stay home instruction: Impacts of the coronavirus pandemic on learner performance for an introductory computer science course. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 294–294, 2020.

[17] M Gabriela Torres, Claire Buck, and Cary Gouldin. Making the leap from the traditional to the virtual educational experience. *New England Journal of Higher Education*, 2020.

[18] Tuul Triyason, Anuchart Tassanaviboon, and Prasert Kanthamanon. Hybrid classroom: Designing for the new normal after covid-19 pandemic. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–8, 2020.

[19] Brad Wuetherick. Basics of qualitative research: Techniques and procedures for developing grounded theory. *Canadian Journal of University Continuing Education*, 36(2), 2010.

Teaching a Penetration Testing Course during COVID-19 - Lessons Learned*

Mohamed Lotfy
Utah Valley University
Orem, UT 84058
MohamedL@uvu.edu

Abstract

Teaching penetration testing is becoming integral in cybersecurity education. In this paper, the structure and design of a senior-level competency-based penetration testing course that had to be offered online due to COVID-19 are presented. Lessons learned from teaching the course online and requiring students to build their own virtual environment are shared. Student course evaluation and what helped them to learn the most are presented and discussed. Analysis of student overall course evaluation responses showed that despite the course offerings were online, the course content, having a personal virtual environment, video facilitated hands-on experience with common penetration testing tools, detailed assignment feedback, and timely faculty responsiveness enabled students to acquire the different offensive cybersecurity skills.

1 Introduction

The computing field is moving from knowledge-based to competency-based computing education. According to IT2017 and CC2020 curricula guidelines, competency-based computing education should connect the knowledge (know-what), skills (know-how), and dispositions (know-why) dimensions regarding a specific context to accomplish a task[7, 2]. "In today's world, graduates must be able to perform in the workplace with appropriate technical skills and human qualities in addition to subject knowledge". Computing degree programs

should be organized around the knowledge, skills, and dispositions dimensions to enable student's career readiness. The security technology and integration knowledge area is considered of medium to high importance in the computer science (CS), cybersecurity (CSEC), information technology (IT), and software engineering (SE) programs[2]. In addition, IT2017 and CC2020 identified "Implement systems, apply tools, and use concepts to minimize the risk to an organization's cyberspace to address cybersecurity threats. (Tools and threats)" as a required IT competency in the ITE-CSP cybersecurity principles section.

In recent years, teaching offensive security in CS, information systems (IS), IT, and CSEC programs became central in cybersecurity education[1, 8]. Offensive Security, through penetration testing/ethical hacking courses, allows students to gain the needed knowledge of how systems and data networks are built and maintained, and learn the needed skills of how to attack and access them using current tools[5, 3, 4]. Connecting the systems and data networks knowledge with how to perform penetration testing skills, and thinking like a hacker within a specific context enables students to develop and gain cybersecurity competency. In addition, the penetration testing/ethical hacking hands-on assignments let students develop the needed cybersecurity capabilities thus enabling them later to build layered defenses that harden the systems to penetration.

Penetration testing phases include information gathering through passive reconnaissance, systems and data networks scanning, vulnerability analysis, exploitation, and post-exploitation. Open-source tools are used in the different phases. Teaching penetration testing requires an attacking host that is used to perform the different penetration testing phases on different vulnerable hosts. Using an encapsulated environment where the different attacks on vulnerable hosts can be conducted, reduces the risk to institutional networks and systems. To allow students to gain hands-on penetration testing experience, some institutions use on-campus cybersecurity labs that use a segmented network while other institutions use a cloud-based lab environment[5, 9, 10].

At Utah Valley University, the IT3700 Information Security–Network Defense and Countermeasures competency-based cybersecurity course has been taught face-to-face in a 16-week semester format. The course examines advanced information security concepts through an applied viewpoint using a hands-on application of real-world techniques and the use of current cybersecurity software. Students attending the course are seniors in the IS and IT programs. Due to the COVID-19 pandemic, all face-to-face offerings in the fall 2020 and spring 2021 semesters moved to online delivery using the Canvas learning management systems. In the face-to-face delivery, on-campus labs were used to conduct the hands-on assignments. Due to the online delivery, students had to install VMWare Workstation on their laptops or PCs and build

a virtual environment to complete the required assignments/labs.

In the remaining sections, the structure of the Network Defense and Countermeasures online course as well as the course assignments and the used tools are presented. Student evaluation and feedback of the course are shared followed by a discussion on the findings and lessons learned from offering the course 100% online.

2 Course Structure

The delivery of the online IT3700 course involved video recordings, presentations, demonstrations, hands-on learning activities, and assignments. The course Canvas shell included the course information, course syllabus, grade book, calendar, course materials/modules, video lectures, assignments, labs, exams, quizzes, a discussion forum, etc. Quizzes were given on each topic covered and completed in the Canvas course shell. A midterm exam and a comprehensive final exam covering the assigned readings and labs were also completed in the Canvas course shell. Table 1 provides the course grading weights.

Table 1: Course Grading Weights

Hands-on Labs	65%
Quizzes	10%
Exams	25%

To conduct the hands-on assignments, students were required to install VMWare Workstation Pro, offered free through VMware Academic Software Licensing Program with the university, on their laptops or PCs and build a virtual environment. The virtual environment included an Offensive Security Kali Linux, textbook author's Ubuntu Linux, Metasploitable 2 Linux by Rapid7, a customized Windows XP, CentOS server, Security Onion 16.04.7.1, and Telekom-security tpotce — a honeypot. Table 2 shows the structure and software/tools used in the course.

In the first two-thirds of the course, students performed different tasks in the penetration testing phases. In the last third of the course, students implemented defensive strategies to secure the hosts and servers. The hands-on assignments are constructed to allow the student to use and apply the knowledge from the readings, videos, online resources, etc., to develop the penetration testing skills within different scenarios/contexts.

Each assignment, provided in the assignment module/page, was very detailed with descriptions and instructions to enable the student to perform the different tasks. Each assignment module/page included four areas, the pur-

Table 2: Course Weekly Structure

Week	Hands-on Assignment	Software/Tools used
1	Verizon Data Breach Report Cybercrime Laws	Setup VMWare environment Install Kali Linux
2	Information Gathering and Passive Reconnaissance Report	Maltego, Shodan, theHarvester, dig, whois, Netcraft, MetaGooFil
3	Port and OS scanning Scan your Home Network	Nmap and Zenmap
4	Port and OS scanning Packet capture	Nmap WireShark
5	Vulnerability Assessment	Nessus
6	Enumeration	rpcclient, enum4linux, and smbclient
7	Exploitation	Nessus, Nmap, Metasploit Meterpreter
8	Social Engineering	Software Engineering toolkit (SET)
9	Passwords Hash Gathering	Nmap, netcat Metasploit Meterpreter
10	Password Cracking	John the Ripper, HashCat and ophcrack
11	Server Hardening	CentOS VM and Nmap
12	HIDS System Installation ICMP packet detection-snort rules	Security Onion VM hping3 and Squill
13	Honeypot server Installation	tpotce VM
14	HIDS TCP Packet Detection Xmas, SYN, FIN, Null-snort rules	Security Onion VM hping3 and Squill
15	Attacking a Honeypot detecting the different attacks on the GUI dashboard	Nmap, Nessus, Metasploit, tpotce Kibana

pose and goals of the assignment, the needed VMs and software tools, the tasks that should be performed, and the expected deliverables. The deliverables were written documents or technical reports showing screenshots of accomplished tasks with reflection on the lesson learned. To enable students to acquire the needed competency and achieve the course outcomes, the faculty provided detailed feedback on each graded assignment. The provided feedback explained what the student did well, what did the student miss, how the student used the tools to meet the assignment/lab requirements, any additional resources or tools that should have been used, and how the assignment fits in the penetration testing phases.

To keep students on track, announcements were posted at the beginning of each week detailing all the needed work — readings, quizzes, and assignments. Also, recorded videos showing why and how to use the different tools to conduct the penetration testing tasks were posted weekly in the announcements as well as the course media folder. The faculty conducted student hours for two hours on three different weekdays using the course Microsoft Teams channel. In addition, student emails during the weekends were answered in less than 24 hours.

A Q&A discussion forum was available for the students to ask questions and clarifications about assignments, course materials, and/or assessments. Students were encouraged to use the course MS Teams channel and the weekly discussion Q&A forum to answer each other's questions and provide help if it is not a quiz or exam-related. In addition, the course calendar was populated with all the assignments and their due dates.

3 Student Course Evaluation and Feedback

At the end of each course, students were provided an online course evaluation form. Table 3 shows the results for the IT3700 Fall 2020 and Spring 2021 overall course evaluation. The same faculty taught the courses and graded all the assignments.

Table 3: Overall Course Evaluation Results

	SA(%)	A(%)	N(%)	D(%)	SD(%)	Avg	StdDev
Fall 2020 (N=23)							
Q1	61	30	9			4.52	0.65
Q2	57	30	9	4		4.39	0.826
Q3	55	36	9			4.45	0.666
Q4	61	30	9			4.52	0.656
Spring 2021 (N=28)							
Q1	61	29	7		4	4.43	0.90
Q2	64	21	11		4	4.43	0.94
Q3	57	25	14		4	4.32	0.97
Q4	57	25	14		4	4.32	0.97

The overall course evaluation area used a five-point Likert scale to answer the following questions:

Q1: I learned more about the subject as a result of taking this class.
Q2: I learned how this subject can be used to address issues outside of the classroom.
Q3: This class challenged me to think in new ways.
Q4: I developed one or more essential skills as a result of this class.

Students were also given the chance to answer the following open-ended question, "What helped you learn the most?". The following were the written responses provided by the students who opted to answer the open-ended question. Each bullet represents the whole received response from each student.

Fall 2020

- "Having my own virtual environments to play around in."
- "The lab videos were a great resource to have."
- "I learned a lot about network traffic in this class and ports, packets, etc. The 'exploit' assignments were very helpful to me."
- "Just being able to do hands on work."
- "Once we started having video lectures, the class got a lot easier to understand."
- "Professor was very helpful, he would made videos walking us through each assignments which was great.He was always responding to emails on time and would give you feedback after you complete each homework. Great Professor."
- "The extra walk-through videos really helped me along."
- "The hands-on with software and videos that demonstrated how the software was being used."
- "The recordings of the tech used in the class."
- "The videos because that class was online."
- "Watching the videos provided by the professor to explain what was going on in the assignments."
- "Posted video walk-throughs helped a lot."
- "Video walk-throughs."

Spring 2021

- "Being shown the way with the challenge to go beyond the scope of the assignment, that little bit of extra effort."
- "Doing the labs."
- "The content itself was very valuable and I learned that way."
- "The feedback I received after each assignment was very helpful and insightful."
- "The lecture videos, book, assignments, and other sources helped me understand and learn the material The feedback after every assignment was incredibly helpful as well! Not many instructors give feedback. Great class overall. Learned a lot of new things! Thank you for the awesome semester."
- "The structure of the assignments combined with the high quality narration/ explanations as well as the professor responsiveness to questions helped the most."
- "The videos were pretty helpful in understanding how to do the labs."
- "The videos were very helpful."
- "The walkthrough videos."
- "Watching the videos posted and trying to follow along helped me understand the most. Visualizing what the professor is talking about at the same time helps me learn the most."

- "In class demonstrations and lectures."
- "Practical lessons and video walk though on how to do it. Good feedback on all graded assignments."
- "The online research for commands."
- "Video tutorials."

4 Discussion

The submitted graded student work showed that the students applied the learned knowledge and gained the needed skills to perform the penetration testing phases in different scenarios/contexts demonstrating the use of common offensive security tools. Also, the student graded work showed that the students understood why each tool was used and why the tool usage fit the penetration testing task within the provided scenario/context.

Student response to the course evaluation questions showed that 91% of the students in the fall of 2020 and 82% of the students in spring 2021 developed one or more essential penetration testing skills. Also, 91% of the students in the fall of 2020 and 90% of the students in spring 2021 learned more about penetration testing as a result of taking the course. Students learned how penetration testing can be used to address outside the class issues, 87% in the fall of 2020 and 85% in the spring of 2021. Lastly, 91% of the students in the fall of 2020 and 82% of the students in spring 2021 agreed that the course challenged them to think in new ways.

All students managed to install VMware Workstation Pro on their laptops and PCs, set up the required VMs on NAT, and complete all the assignments/labs. Student written comments, in section 3 above answering the open-ended question, showed that the hands-on experience in their own virtual environment enabled them to learn the different offensive cybersecurity skills, which is similar to what [4, 6, 10] had found. Some students had to reduce the VMs allocated RAM size and number of CPUs which slowed their VMs response. The Security Onion IDS and tpotce honeypot VMs ran many visualization services to capture and display the elastic stack data, which for some students slowed their systems to a halt. After stopping most of the unneeded visualization services, the impacted students managed to complete the Security Onion and tpotce assignments/labs.

Most students responding to the open-ended question mentioned that the weekly provided walk-through videos helped them to learn the most. Some students highlighted the value of the detailed feedback on the graded assignments on their learning. In addition, some students found the faculty's quick responsiveness an enabler to learning. The practicality of the penetration testing course content was another element that allowed students to learn the most.

The following is a summary of the lessons learned:

- Competency-based penetration testing/ethical hacking courses can be offered 100% online allowing students to gain and acquire the needed knowledge and skills.
- The hands-on experience in the students' own virtual environment enabled them to learn the different offensive cybersecurity skills and how to apply them.
- Weekly detailed walk-through videos showing how to use the different penetration testing tools facilitated student learning.
- Detailed assignment/lab instructions coupled with detailed feedback included in the graded assignment increased student learning.
- Timely faculty response to student questions and emails is important to support student learning in online courses.

5 Conclusions

Penetration testing/ethical hacking courses provide advanced cybersecurity and information security concepts through an applied viewpoint using a hands-on application of real-world techniques using current cybersecurity software and tools are an integral component in current computing education. In this paper, we provided the structure of a penetration testing course that had to be offered online instead of face-to-face due to the COVID-19 pandemic. Learned experiences and student overall evaluation of the two online iterations of the course were shared. Analysis of student overall course evaluation and qualitative responses showed that despite the course offerings were online, the course content, having a personal virtual environment, hands-on experience with common penetration testing tools facilitated with walk-through videos, detailed faculty feedback, and timely faculty responsiveness enabled students to acquire the different offensive cybersecurity skills.

References

[1] Miriam E Armstrong, Keith S Jones, Akbar Siami Namin, and David C Newton. Knowledge, skills, and abilities for specialized curricula in cyber defense: Results from interviews with cyber professionals. *ACM Transactions on Computing Education (TOCE)*, 20(4):1–25, 2020.

[2] CC2020 Task Force. *Computing Curricula 2020: Paradigms for Global Computing Education*. Association for Computing Machinery, New York, NY, USA, 2020.

[3] Regina Hartley, Dawn Medlin, and Zach Houlik. Ethical hacking: Educating future cybersecurity professionals. In *Proceedings of the EDSIG Conference ISSN*, volume 2473, page 3857, 2017.

[4] Chengcheng Li. Penetration testing curriculum development in practice. *Journal of Information Technology Education: Innovations in Practice*, 14(1):85–99, 2015.

[5] Lionel Mew. The information security undergraduate curriculum: Evolution of a small program. In *Proceedings of the EDSIG Conference ISSN*, volume 2473, page 3857, 2016.

[6] Arghir-Nicolae Moldovan and Ioana Ghergulescu. Leveraging virtual labs for personalised group-based assessment in a postgraduate network security and penetration testing module. In *2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA*, pages 1–6. IEEE, 2020.

[7] Task Group on Information Technology Curricula. *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. Association for Computing Machinery, New York, NY, USA, 2017.

[8] Yang Wang, Margaret McCoey, and Qian Hu. Developing an undergraduate course curriculum for ethical hacking. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 330–335, 2020.

[9] Yien Wang and Jianhua Yang. Ethical hacking and network defense: choose your best network vulnerability scanning tool. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 110–113. IEEE, 2017.

[10] Xiaodong Yue and Hyungbae Park. Design of virtual labs for an ethical hacking course. *Journal of Computing Sciences in Colleges*, 35(6):31–38, 2020.

Autonomy-Supportive Game Benefits Both Inexperienced and Experienced Programmers*

Michael J. Lee and Ruiqi Shen
Department of Informatics
New Jersey Institute of Technology
Newark, NJ 07102
{mjlee, rs858}@njit.edu

Abstract

As more people turn to discretionary online tools to learn new skills such as computer programming, exploring how to better support a wide range of learners is becoming increasingly essential to train the next generation of highly skilled technology workers. In our prior work, users with high learner autonomy complained that most online resources they used to learn more programming did not provide them with the flexibility they preferred to navigate through learning materials, locking them into a set sequence of topics/concepts. To explore this, we implemented a *level-jumping* feature into an online educational programming game. We tested it with 350 new users, tracking their progress through the game for 7 days each. We found that those with high learner autonomy did use the level jumping feature more than those with low learner autonomy. We also found that males were more likely to use this new feature, regardless of learner autonomy level, compared to their female counterparts. Finally, we found that those with low learner autonomy ultimately completed more levels than their high autonomy counterparts, and that this was particularly true of female learners (who completed the most levels overall). Based on these findings, we believe that autonomous-supportive features such as flexible navigation may be beneficial to all users of online educational tools, and that encouraging its use by a wider group of users (particularly females), may increase positive effects.

1 Introduction and Related Work

With the continued influx of individuals turning to discretionary online learning learning resources such as Massive Open Online Courses (MOOCs), tutorial sites, and educational games, learning more about how to support a wide range of people with different preferences, experience, and learner autonomy, is essential to train the next generation of highly skilled technology workers. According to educators and researchers, learner autonomy — the ability for one to control their own learning [6]—plays a vital role in developing lifelong learners [3].

In order to gain a better understanding of their autonomy, our past work explored the experience of Computer Science (CS) learners', particularly about learning CS both online and in the classroom [14, 17]. We found that learners who showed a *high level* of autonomy felt that they were not supported by the educational system(s) or teachers. For example, they complained that the online systems they used did not give them enough freedom to explore on their own, and that their teachers often failed to help them achieve their learning goals. On the other hand, we found that learners who showed a *low level* of autonomy felt that they needed extra guidance from their teachers and curriculum. Interview results also indicated some patterns, for example, that learners with more subject-area experience showed higher levels of autonomy than those with less experience, and learners with a higher level of autonomy preferred to study using an autonomy-supportive system while those with lower levels of autonomy preferred to study using a non-autonomy supportive system [13, 17].

Considering these past results and the role learner autonomy plays in developing lifelong learning, we believe that it is important to address the needs of CS learners with different levels of learner autonomy. In particular, we found in prior work [13, 16] that autonomy-supportive features (i.e., the freedom to freely navigate to any portion of a course's curriculum) in learning systems were consistently requested by those with high learner autonomy. This was in contrast to those with low learner autonomy, who preferred much more structured and linear pathways through a curriculum.

In order to test how an autonomy-supportive feature might affect learners (based primarily their level of learning autonomy), we implemented a *level-jumping* feature into an online educational programming game (see Figure 1). This is in contrast to most online learning curriculums and MOOCs that we have encountered, which are often locked to a specific sequence where later parts of the course are inaccessible until earlier parts are completed. We tested the game with this new level jumping feature with 350 new users, tracking their progress through the game for one week (7 days) each, spanning a total of 1.5 months.

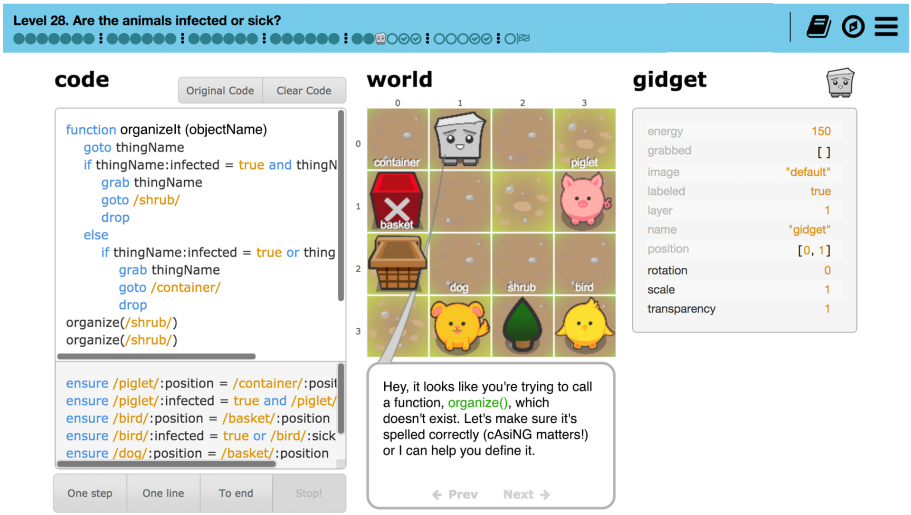


Figure 1: *A screenshot of the Gidget introductory programming game.*

2 Method

2.1 The Gidget Educational Game

We modified our free introductory coding game, Gidget *(www.helpgidget.org)*, for this study. Gidget has a total of 37 core levels in its curriculum, where each level teaches a new programming concept (e.g., variable assignment, conditionals, loops, functions, objects) using a Python-like, imperative language [8, 10]. The objective of each level is to fix existing code to help the game's protagonist pass 1–4 test cases (i.e., statements that evaluate to `true`) after running the code. Each level introduces at least one new programming concept, becoming progressively difficult in subsequent levels. Therefore, users are exposed to more programming concepts the farther they progress through the game. Finally, the game also includes a set of help features to help players overcome obstacles while coding on their own [7, 10]. This includes a frustration detector that provides encouraging hints/messages to those that are struggling with a level [9], and also auto-generates additional levels covering the same concept(s) to provide additional practice [8].

Normally, the game follows a specific order of levels (i.e., curriculum), building on content from previous levels. While the user interface shows the sequence/map of all core levels in the game (and indicating the players' current level; see top of Figure 1), it only allows the player to jump back to any previously completed level (at any time during game play). Players can also jump

forward to the last level they have reached sequentially, but no further. All levels are visualized on the map as circles, with completed levels shown as solid circles, incomplete levels shown as hollow circles, and incomplete exam levels (explained in [10]) shown as hollow circles with a check mark. Finally, the currently loaded level is indicated with the Gidget character (see Figure 2). Hovering the mouse cursor over an incomplete level does not show any visual change, and clicking on an incomplete level does not do anything.

For this study, we modified the level-selection interface to allow players to jump to any level in the game, regardless of level completion status. Placing the mouse cursor on any other level grays out the current level's Gidget character and places a solid Gidget character that slowly rocks back-and-forth on that level marker. Clicking on the rocking character immediately jumps the player to that level. In addition, to keep the overall experience consistent across all users of this study, we disabled the game's auto-generated extra levels (as described in [8]). This was to prevent cases where someone might jump to a difficult level, trigger the frustration detector, then offered multiple additional practice levels covering the same concept(s). Finally, we specifically pointed out this level-jumping feature in the game's introductory on-boarding tutorial (which all players see the first time they load the game), explaining the user interface, interaction method, and the level-jumping feature.

Figure 2: *Closeup of the level selection map. The current level shows Gidget, completed levels are solid circles, uncompleted levels are hollow circles, and uncompleted exam levels are hollow circles with a check mark.*

2.2 Participant Recruitment

Our goal was to observe if and how players would use the level-jumping features within the game. We evaluated our system with a group of 350 new users of the game. The sign-up screen asked users for their age, gender, e-mail address, a checkbox indicating whether they have prior programming experience, and a checkbox (with link to consent form) asking if they were willing to participate in a research experiment. We intentionally did not define "programming" or "programming experience" as we determined in past studies [14, 15] that using a specific definition could potentially confuse or discourage participants who might consequently miscategorize themselves or self-select out of participation even though they meet our eligibility criteria. Mirroring a previous study [14], we asked those who indicated that they had prior programming experience two additional questions: how many years of programming experience they had (rounded up to the nearest .5 or integer), and how they would rate their

programming experience level on a four-level scale (beginner, intermediate, advanced, and professional). We used these two measures to assign each player a learner autonomy score from 1 (low learner autonomy) to 3 (high learner autonomy) based on our prior work [14], which showed that these two measures were significantly correlated with learner autonomy. This prior study combined subsets of the Learner Autonomy Scale created by Macaskill and Taylor [11] and the E-learning Autonomy Scale developed by Firat [5], and demonstrated that more years of experience and higher self-rating in programming experience has a positive relationship with autonomy level.

For this study, we only selected users that indicated they were 18+ years and willing to participate in a research experiment. Adapting the methodology from our prior studies [8, 9], we set the observation time to 7 days (168 hours) per user to have a consistent evaluation window for all users. To promote quick account creation, we did not collect other demographic information such as ethnicity, geographical location, or education level. Participants were required to read and digitally sign an online consent form that briefly described the study. We were intentionally vague in our description of the level-jumping feature, stating that we were "testing new navigational features" to minimize potential leading or biasing of participants to pay attention more to that specific part of the interface. However, we debriefed all participants of the study procedures 7 days after the end of their individual observation window, by e-mail.

3 Results & Discussion

We report on our quantitative results comparing our participants' outcomes—split by demographic and experience features—using nonparametric Wilcoxon rank sums tests, Chi-Squared tests, or simple linear regression, with a confidence of $\alpha = 0.05$, as our our data were not normally distributed. For all post-hoc analyses regarding gender data, we use the Bonferroni correction for three comparisons: ($\alpha = .05/3 = 0.0167$).

The study included 350 participants (aged 18–58; median 20). As a whole, our participants were composed of 180 females (51.4%), 161 males (46%), and 9 'not listed' or 'decline to state' (2.6%). In addition, 255 (72.9%) indicated that they did not have any prior programming experience, and 95 (27.1%) indicated that they had at least .5 years of prior programming experience (latter's range .5–33; median 2). We operationalized our key dependent variables, *engagement* and *jumping*, as the number of levels completed and the number of times the jumping feature was used, respectively.

3.1 High Learner Autonomy Players Use Jumping Feature More

We found that all learners used the level jumping feature at least 2 times, regardless of having low learner autonomy (range 2-17; median 3) or high learner autonomy (range 2-37; median 8). Looking at the data more closely, we found that there was a significant difference in the number of levels participants completed by autonomy level ($W = 26703, Z = 12.370, p < .05$), with the high autonomy learners using the feature more than their counterparts.

We believe that all learners jumped at least two times because this feature was specifically mentioned in the on-boarding game tutorial, and at the minimum, someone using the feature to jump forward (first jump), would need to jump back to their original level (second jump). Next, our finding that high autonomy learners use the jumping feature more often than their low autonomy counterparts verifies our hypothesis (based on our previous work in [14]) that those with more experience (and therefore higher learner autonomy) would use and benefit from this jumping feature. Unlike low autonomy (inexperienced) learners, who do not necessarily know much about the topic and therefore would be better served learning programming concepts in a sequenced curriculum, the goal of high autonomy (experienced) learners may be to review or improve on their existing programming skills, and/or to look for programming resources. Therefore, they may be more likely to use the jumping feature to browse through the different parts of the curriculum quickly, being more in control of their learning.

3.2 Males Use Jumping Feature More

We found a significant difference in usage of the jumping feature by gender (χ^2(2, N=350)=17.226, p<.05). Doing post-hoc analysis with the Bonferroni correction, we found that males used the jumping feature significantly more overall than their female counterparts (W=42.307,Z=4.109,p<.05/3). This result was independent of low learner autonomy (χ^2(2,N=255)=6.1464, p<.05) or high learner autonomy (χ^2(2,N=95)=6.1583, p<.05) in programming.

This result was not too surprising, as prior research [2] has shown that compared to females, males are statistically more likely to use selective information styles (following the first promising information, then potentially backtracking) [12], have lower risk aversion (be less wary of consequences) [4], and more willing to tinker (playfully experiment) [1]. Based on this, we believe that our male players were more likely to use the jumping feature simply because it was available in the interface (and also mentioned in the tutorial).

3.3 Low Autonomy (Female) Learners Complete More Levels

Next, we explored if there was a difference in the number of levels participants completed. This is not a completely fair comparison, as everyone may have encountered levels in a different sequence (with later levels being considerably more difficulty than earlier levels) because of the jumping feature.

We found that low autonomy learners completed significantly more levels compared to their high autonomy counterparts ($W = 15002.5, Z = -1.987, p < .05$). Further analysis revealed that there was a significant different in the number of levels completed by gender within the low autonomy group ($\chi^2(2, N=255)=43.3806$, $p<.05$). A post-hoc analysis with the Bonferroni correction showed that the low autonomy group females completed significantly more levels compared to their male counterparts (W=-61.579,Z=-6.655,$p<.05/3$). We calculated a simple linear regression to predict level completion based on jumping behavior. Within the low autonomy group, we found a positive relationship between these variables ($F(1, 253) = 255.290, p < .05$), $R^2 = .502$). Examining this more closely, we found that that this affect was strongest with females, where females in the low autonomy group who jumped more often completed more levels ($F(1, 144) = 206.433, p < .05$), $R^2 = .589$).

This result supports our hypothesis discussed in Section 3.1. The goal of high autonomy (experienced) learners may be to review or improve on their existing programming skills, and/or to look for programming resources. If high autonomy learners were using the level jumping feature primarily to explore what programming concepts the game curriculum covered, it would explain why they did not necessarily stay to solve/complete those levels. On the other hand, a low autonomy (inexperienced) learner's aim in playing a programming game is more likely to learn new things, and most or all of the programming concepts would be new to them. Therefore, whether or not they jump through the curriculum, less experienced learners have more incentive to complete levels. Perhaps those low autonomy learners that jump around the levels have a better idea of what is coming next (and also gain additional insights from the broken, starting code each level provides), and therefore more successful in completing levels. Most surprisingly, although our female participants were most likely not to use the jumping feature, those that did went on to be the individuals that completed most (or all) of the game levels. Females who did decide to use the jumping feature may have jumped back and forth between levels as a comprehensive information processing problem-solving strategy [2, 12], where they used the jumping feature to preview what was coming up, thereby gathering fairly complete information about the entire system before proceeding.

4 Conclusion

Our findings show that both high and low autonomy learners (particularly males), used the level-jumping feature, with the former using this feature significantly more than the latter. We also found that high autonomy learners tend not to complete the levels they jump to, and that they complete significantly fewer levels overall compared to their low autonomy counterparts. We also found that the few low autonomy female learners who used the jumping feature readily, also ended up completing more levels than any other group. Designers for online resources teaching programming may benefit from allowing all users to skip around and explore the curriculum, instead of locking them into a specific sequence. They may also do well in encouraging more of their learners (especially females) to use these types of jumping features to have them preview and better prepare for what is coming later in the curriculum.

We have several limitations to our study. We recruited participants who opted into a research study while signing up for an educational game. These participants may already have high motivation, and therefore may not be completely representative of the larger population. Next, we asked participants to self-report their years of programming experience and also to rate their own programming expertise. Participants may have different criterion for these selections and therefore may have led to inconsistencies in our user groupings. The groupings themselves may not account for all the different nuances of experience and/or learner autonomy. For future work, we could use more objective measures such as quizzes to test the skill level of participants as an alternative measure to experience. In addition, we could use pre-post tests to measure how this new jumping feature affects players' learning outcomes, and collect qualitative data from participants through questionnaire or interviews.

Our study results show that both high autonomy and low autonomy learners use the level-jumping feature (presumably to preview levels), and that although low autonomy users are less likely to utilize this feature, those that do are especially successful in completing more levels (particularly females). Our future work will examine these outcomes in more detail, and gather complementary qualitative data, to isolate the features that are causing these effects.

5 Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under grants DRL-1837489 and IIS-1657160. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the NSF or other parties.

References

[1] Laura Beckwith, Cory Kissinger, Margaret Burnett, Susan Wiedenbeck, Joseph Lawrance, Alan Blackwell, and Curtis Cook. Tinkering and gender in end-user programmers' debugging. In *ACM CHI*, pages 231–240, 2006.

[2] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Anicia Peters, and William Jernigan. Gendermag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, 28(6):760–787, 2016.

[3] Philip C Candy. *Self-Direction for Lifelong Learning. A Comprehensive Guide to Theory and Practice.* Eric, 1991.

[4] Thomas Dohmen, Armin Falk, David Huffman, Uwe Sunde, Jürgen Schupp, and Gert G Wagner. Individual risk attitudes: Measurement, determinants, and behavioral consequences. *J. of the European Econ. Assoc.*, 9(3):522–550, 2011.

[5] Mehmet Firat. Measuring the e-learning autonomy of distance education students. *Open Praxis*, 8(3):191–201, 2016.

[6] Henri Holec. *Autonomy and foreign language learning.* Eric, 1979.

[7] Michael J Lee. How can a social debugging game effectively teach computer programming concepts? In *ACM ICER*, pages 181–182, 2013.

[8] Michael J Lee. Auto-generated game levels increase novice programmers' engagement. *The Journal of Computing Sciences in Colleges*, 36(3), 2020.

[9] Michael J Lee. (re)engaging novice online learners in an educational programming game. *The Journal of Computing Sciences in Colleges*, 35(8), 2020.

[10] Michael J Lee, Amy J Ko, and Irwin Kwan. In-game assessments increase novice programmers' engagement and level completion speed. In *ACM ICER*, 2013.

[11] Ann Macaskill and Elissa Taylor. The development of a brief measure of learner autonomy in univ. students. *Studies in Higher Education*, 35(3):351–359, 2010.

[12] Joan Meyers-Levy and Barbara Loken. Revisiting gender differences: What we know and what lies ahead. *J. of Consumer Psychology*, 25(1):129–149, 2015.

[13] Ruiqi Shen. Interactive computer tutors as a programming educator: Improving learners' experiences. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–2. IEEE, 2020.

[14] Ruiqi Shen, Joseph Chiou, and Michael J Lee. Becoming lifelong learners: Cs learners' autonomy. *J. of Computing Sciences in Colleges*, 35(8):267–267, 2020.

[15] Ruiqi Shen and Michael J Lee. Learners' perspectives on learning programming from interactive computer tutors in a mooc. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–5. IEEE, 2020.

[16] Ruiqi Shen, Donghee Wohn, and Michael Lee. Programming learners' perceptions of interactive computer tutors and human teachers. *International Journal of Emerging Technologies in Learning (iJET)*, 15(9):123–142, 2020.

[17] Ruiqi Shen, Donghee Yvette Wohn, and Michael J Lee. Comparison of learning programming between interactive computer tutors and human teachers. In *ACM Conference on Global Computing Education (CompEd)*, pages 2–8, 2019.

Redesign of an Elective Introductory Artificial Intelligence Course in a Credit-Limited Computer Science Curriculum*

George Thomas
Computer Science Department
University of Wisconsin Oshkosh
Oshkosh, WI 54901
thomasg@uwosh.edu

Abstract

Teaching introductory Artificial Intelligence (AI) courses in undergraduate institutions presents a unique set of challenges due to the ever increasing content areas of AI. The goal of an introductory course is to provide a broad introduction to the field, but in an undergraduate setting where this may be the only AI course a student encounters, equipping students with tangible AI skills to apply in their careers after graduation is also highly desirable. This paper presents an experience report that discusses the rationale and methodology used to redesign a standard, introductory course into a more versatile version, and provides high level student evaluation data over an eight year period to assess the efficacy of this course redesign. The key idea in this redesign is to allow a significant portion of the content area to vary from one course offering to another.

1 Introduction

Artificial Intelligence (AI) is an important and dynamic field of computer science. Most computer science programs offer either required or elective courses

in this area. The field of AI is one of the broadest in computer science and encompasses topics as diverse as classical AI search, symbolic and probabilistic reasoning, machine learning, and multiagent systems, to varied application areas such as Natural Language Processing (NLP), robotics, computer vision, etc. Covering all of these areas in a single semester course is impossible, and generally each of these areas warrant their own dedicated course. This creates a dilemma for the instructor when deciding the breadth and depth of topic coverage. This paper describes the experiences at the University of Wisconsin Oshkosh (UWO) in redesigning the AI course taught there.

Due to the vast content area of AI, an ideal coverage of it would require a multi-course sequence rather than a single introductory course. Typical AI sequences, in addition to a broad introductory course, would include courses on machine learning, and specific application areas. Most undergraduate programs do not have the enrollment to fill all these various courses, nor do they have the credits to devote to entire AI course sequences in a standard computer science major. Undergraduate colleges generally offer AI as a single course, with the number of credits devoted to it generally being three credits. There is usually not enough enrollment for a higher level AI elective, or perhaps the program has too many credits allocated to other courses, without instructor coverage for these higher level AI courses available. Yet it is universally acknowledged that AI plays an increasingly important part in student careers after graduation, especially with deep learning currently being seen as a possible panacea to most complex computing problems in industry.

So, the question we tried to address in our course redesign was how we can design an introductory course in AI that gives students not only the broad exposure to the field but also the specific and relevant knowledge that allows them to use AI techniques in their jobs right after graduation.

2 Background

There is much literature devoted to the study of AI courses in the EAAI, SIGCSE and ITiCSE conferences, and a few references in the Journal of Computing in Small Colleges.

Some courses focus on a specific area of AI[1], or on a different student audience such as those in middle school[3], or non-computer science major students[2]. Others focus on using a specific area such as Games to teach AI [4]. But to the best of our knowledge, none attempt to redesign an introductory AI course in the manner we describe.

3 Methodology

First, UWO offers only one course in AI that is an optional elective in the Computer Science (CS) major, which requires approximately 52 major credits, with 12 of those credits being electives. The AI course has generally been very popular among students, but limited teaching credits preclude adding additional, more advanced AI courses, beyond this introductory course. The old version of the introductory AI course, offered through 2012, had its course description as follows:

> *"This is an introductory course in Artificial Intelligence. A tentative list of the topics we will cover, in varying degrees of detail, include: Introductory overview of AI, agents and state spaces; Uninformed and informed search; Constraint satisfaction problems; Adversarial search; Introduction to logic, propositional and predicate calculus; Machine learning including neural networks, decision trees and naïve Bayes learning algorithms."*

The course focused roughly a third of the topic coverage on classical search, a third on logic, and the last third on some machine learning and other assorted topics. The prerequisites for this course were essentially only a CS 2 course. This meant that students might not have seen either propositional or predicate calculus (generally covered in a discrete mathematics course) or even Data Structures. While this prerequisite structure increased the course accessibility to students fairly early in their study, a considerable amount of course contact time was spent in reviewing these fundamentals.

Overall instructor experience in teaching this course was that the logic component of the course, with time spent on introducing propositional and predicate calculus, in addition to covering symbolic logic was not an effective use of course time. Considerable time was also spent on classical search explaining the fundamentals of Data Structures. Consequently, two major changes were implemented in 2013.

First, the prerequisite structure was changed to now require Data Structures. This allowed the classical search coverage to proceed much more effectively. Discrete Mathematics was still not required, so students could still take AI without having completed that course. This preserved some measure of student course accessibility early in their plan of study.

Second, a reexamination of what were the "essential" topics of AI that belong in an introductory course was conducted, and the determination was that the AI course should be composed of the following three components:

1. **Classical Search:** Any introductory course should lay out the challenges that AI faced at its onset and the evolution it went through. Conse-

quently, the topics covered here include uninformed and informed search, local search, game search (including Expectiminimax and Monte Carlo search) and constraint satisfaction. Details of the algorithms involved and programming projects that involve solving application problems using these algorithms are covered here.

2. **Machine Learning:** A third of the course is devoted to covering Machine learning, recognizing its importance in every domain today. While this would normally be covered in a course devoted to machine learning, our restriction in having a single AI course required the significant coverage of machine learning in this introductory course. Topics covered include linear regression, decision trees, Bayesian learning and neural networks, including some coverage of deep learning, ensemble methods, and evaluation of learning algorithms. This ensures that students have a deep enough understanding of machine learning to use these techniques right away in their careers after graduation. The degree of coverage of deep learning has varied but there is not enough time to cover all key areas such as convolutional neural networks, recurrent neural networks and generative adversarial networks, so the instructor has to select what would be covered in a specific course offering.

3. **One application area of AI:** Taking into account the highly dynamic nature of AI, a third of the course content is loosely focused on an "application area" of AI and is left up to the instructor to decide specifically what that might be. Choices could include Natural language processing, Computer Vision, Multiagent Systems, Planning including Markov decision processes, and symbolic methods in AI. The decision to focus on one area rather than provide a broad introduction to many different areas was so that students can see in-depth examples and applications of AI rather than receive just a broad overall knowledge. It is expected that students will do 1-2 programming projects in the chosen area. At UWO, the instructors have chosen to focus on NLP but the course description allows maximum flexibility from one offering to the next.

   The exclusion of symbolic logic from the standard topics might be controversial but it is a recognition that including this topic in course coverage will occupy a third of the course contact time due to the lack of prerequisite knowledge required from the students. It is also a recognition that deep learning is currently driving research in AI, especially over symbolic approaches. But it also provides an instructor the opportunity to cover symbolic logic in depth if they should so choose.

The course description for this redesigned AI course is given below:

> *"This course is an introduction to the field of artificial intelligence. A survey of classical search in artificial intelligence and machine learning, and an in-depth examination of a specific application area such as robotics, theorem proving, computer vision, natural language processing, etc. are covered. Students are expected to demonstrate mastery via computer programs using the techniques of artificial intelligence."*

The course outcomes list specific learning outcomes for the first two areas, as would be expected in a standard introductory AI course. The course outcomes for the "application area" had to be phrased in a generic manner. These outcomes are included here for illustration:

> *With respect to ONE selected "big application" area of AI such as Reasoning and Theorem Proving, Natural Language Processing, Computer Vision, Robotics, Multiagent Systems, etc:*
>
> *(a) identify and describe the motivation, terminology, foundations and key concepts in the selected area*
> *(b) describe and manually trace the essential algorithms in the selected area*
> *(c) use available libraries and software to test the key algorithms and extend them to specific application projects in the selected area*

We believe such an outcome formulation provides maximum flexibility to the instructor while giving students a clear understanding of what they will be able to demonstrate at the end of the course.

4 Evaluation

While we do not have objective data collected, with appropriate controls enforced, we have subjective data inferred from student evaluation data and the instructor's experience.

The AI course is an elective taught every three semesters. We present data for five offerings of the course over a seven year span. The same instructor taught the course these five times. The course was taught by a different instructor in 2012, for which data is not available and student evaluation data was not collected in the spring 2020 semester due to the pandemic. The 2011 offering was based on the old AI course, and the 2014 and subsequent offerings are based on the redesigned course.

First, the instructor noticed a marked difference in student interest in the redesigned version of the course. NLP was used as the application area in all offerings of this course. Using NLP as the application area piqued student interest, and the prerequisite change ensured students were better prepared. A side benefit of the ability to focus on a specific AI application area through a "mini-course" was that three students were interested enough in the application area to work on subsequent practicums with the instructor. One of those practicums resulted in a peer reviewed publication.

The overall student experience seems to correlate with the instructor's assessment. Table 1 shows the relevant questions dealing with course content.

Table 1: Student Evaluation Data

Criteria	2011	2014	2015	2017	2018
Q1: Course organization	4.2	4.28	4.53	4.59	4.36
Q2: Value of lectures	3.8	4.06	4.42	4.35	4.24
Q9: Effective instructor	3.8	4.11	4.26	4.65	4.24
Q11: Appropriate prerequisites	3.14	4.19	4.11	4.31	4.26
Q12: Recommend to others	3.6	3.89	4.05	4.31	4.08
Q13: Course difficulty	2.57	2.72	2.47	2.65	2.71
Q14: Course pace	2.95	3.22	3.32	3.18	3.24
Q15: Amount of work	2.85	2.72	2.74	2.94	2.88
Total Respondents	21	18	19	17	26
Total Enrollment	27	22	22	18	27

Questions Q1-Q12 were rated on a 1-5 scale (with 1 being *strongly disagree* and 5 being *strongly agree*). As can be seen, the value of lectures and overall instructor effectiveness increase significantly in the redesigned course, over the 2011 old version of the course. While this is possibly due to the instructor getting more experience in teaching the course, we also strongly believe the redesign helped significantly. There was also a marked increase in the appropriateness of the prerequisites, and the degree to which students would recommend the course to others.

Questions 13-15 were rated on a scale of *hard* to *easy*, *slow* to *fast* and *too much* to *too little* respectively, with a score of 3 being *about right*. Students perceive the course pace to have increased but, interestingly, the course difficulty and the amount of work required seem to be mostly the same. This would seem to indicate that this increased pace is appropriate for the redesigned course.

5 Conclusion

We present an outline of the redesign of our introductory AI course. Our experience is based on high level student evaluation data with one instructor over five course offerings, and this small sample size is a limitation of our study.

Our redesign incorporates a "mini-course" in an AI application that would typically be its own upper level course in a program that can afford to do so. Our experience has been that we can cover the application area in the equivalent of one credit hour (or roughly a third of the overall course time) to a depth that allows students to be productive in programming applications of significant complexity. The preceding two credit hours devoted to the basics of AI, and machine learning seem to set the stage for this higher degree of student comprehension in the application area. Student evaluation data seem to indicate that students prefer this redesigned course. And such a course organization affords a high degree of flexibility to the instructor.

References

[1] Travis Mandel and Jens Mache. Developing a short undergraduate introduction to online machine learning. *The Journal of Computing Sciences in Colleges*, 322(1), 2016.

[2] Matthew Merzbacher. Open artificial intelligence - one course for all. In *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, SIGCSE 01, pages 110–113, 2001.

[3] Alpay Sabuncuoglu. Designing one year curriculum to teach artificial intelligence for middle school. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 20, pages 96–102, 2020.

[4] Daniel Wong, Ryan Zink, and Sven Koenig. Teaching artificial intelligence and robotics via games. In *Proceedings of the AAAI 2010 Symposium on Educational Advances in Artificial Intelligence*, EAAI 10, 2010.

Creating Hands-on Assignments to Teach Symmetric Encryption with Increased Student Involvement*

Jonathan Neilan and Sayeed Sajal
Department of Computer Science
Utah Valley University
Orem, UT 84058
{jonathan.neilan, sayeed.sajal}@uvu.edu

Abstract

Encryption is a ubiquitous and necessary aspect of our online world. An education in computer science would be incomplete without at least a high-level understanding of encryption algorithms. We look into common paradigms for symmetric encryption in order to create educational, application-based assignments for testing competency in symmetric encryption concepts. We first look into Data Encryption Standard (DES) for an introductory view of why certain cryptographic qualities matter and how they are achieved and create an assignment to effectively implement DES. next, we looked into Advanced Encryption Standard (AES) and the other top entries of the NIST competition from which AES was selected in order to observe multiple examples of implemented block ciphers. We created an assignment to implement a block cipher, the specifics of which are of the student's choosing in order to provide leeway for each student to brainstorm and implement their own ideas.

1 Introduction

Symmetric encryption is the model of encrypting data by using a single, shared secret key that is used for both encrypting and decrypting data. The prominent predecessor for many modern day symmetric encryption algorithms is

DES, Data Encryption Standard. DES has long been deprecated, no longer secure for any serious web traffic but still presents a good initial point of study for cryptography. DES uses 64-bit blocks of plain text data and a 64-bit key as input; a total of sixteen Feistel rounds are performed to create cipher text[6]. A single Feistel round entails splitting the plain text input in half, performing an encryption step on one of the halves, and then swapping the 'left' and 'right' halves. A different key, referred to as a round key, must be used at each Feistel round. These round keys are generated from the main key according to a defined "key schedule" which includes performing left shifts and permutations[6].

The National Institute of Standards and Technology (NIST) held a competition to find a new encryption standard. The chosen winner "Rijndael" would become the new AES, Advanced Encryption Standard. It is still the current standard for symmetric encryption algorithms and is the most popular cipher used in both public and private sectors. Rijndael's block cipher guarantees "high diffusion over multiple rounds" through three steps - byte substitution, shift row, mix column, and finally adding the round key[5]. It provides the best mix of defense against well-known attacks as well as accomplishing the "avalanche effect"; a property of algorithms where a small, single change in an input yields large changes in the output.

The top five finalists of NIST's encryption competition were Rijndael (the winner), Serpent, Twofish, RC6, and MARS. All five of these finalists used one of two forms of block ciphers: Substitution-permutation ciphers or Feistel network ciphers.

Substitution-permutation ciphers work by applying alternating rounds of substitution boxes and permutation boxes to the input plaintext and key. Feistel ciphers involve splitting an input into two halves and performing a non-linear function to only the right half. The result is also "added into the left half, and subsequently left and right halves are swapped[8]".

2 Background

Initial research we performed on symmetric cryptography assignments yielded varied results. There were security-based assignments aimed at teaching students the weaknesses of DES and how they are exploitable. There were fairly simple exercises which were meant to teach steps of an encryption process as part of a whole, such as one that provided full AES functionality and only required students to perform the base-64 encoding of messages before and after the encryption and decryption process between two clients. A more mathematically inclined one involved analyzing block cipher chaining and cryptanalysis of various paradigms. An assignment from a Utah Valley University course in

2019 goes over encryption concepts and application but involves no programming. Another course from 2020 that did require encryption programming also required the use of key generators and libraries to complete the exercise. Completion of the assignment also required construction of an API for a client and server to communicate with each other via a REST API.

The languages used by these examples were mostly C++ and Python, their justifications being that they both allow for the easy use of external libraries such as Key generators. Those using C++ further cited that as a low-level language it has quick access to the hardware, a desirable quality for performing ciphers. Those using Python justified their approach as an easier way to educate students on new concepts.

3 Project Details

Based on our research into previous assignments, pedagogical methods, and paradigms we wished our students to focus on, we decided the assignments would entail two parts. The first is the implementation of a specific, well-known algorithm such as DES for first learning cryptography concepts. The second is to allow students more leeway in what they create by providing a general paradigm common to encryption algorithms to serve as a model to follow. The model that was chosen was block ciphers. Our reasoning will be provided below.

3.1 Tools and Languages

The language the programming assignments were created in was C++ (standard C++11/C++14 compliant). This provided the best balance of being "low"/close to the machine and being a popular language in which the assignment could be expounded upon in future edits. The software was created in Microsoft Visual Studio 2017 (v141) on a Windows 10 Operating System.

3.2 Assignment 1 - DES

Given its lengthy existence and cryptanalysis that has been performed on it, implementation of DES as an assignment provides a good introduction to the types and qualities of cipher "rounds" that encryption functions may perform. The assignment format is as follows: the steps of DES are broken up into their own functions. The steps that require "boxes", which are the expansion step, substitution step, and permutation step, have each fixed "box" provided as multi-dimensional arrays, seen in Figure 1, for use in their respective steps in the encryption process.

Comments are provided to explain the role that each function and step plays in the encryption process. As an example, explaining S-box's role is based on documentation for the fixed-sized array(s), "A straightforward way to implement simple nonlinear functions are lookup tables or S-boxes. The DES

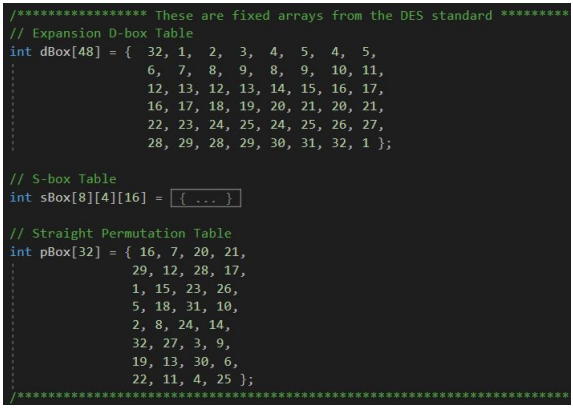


Figure 1: D-box, S-box, and P-box (S-box kept collapsed due to size)

uses eight different S-boxes with six input bits and four output bits (denoted with 6 to 4)"[8]. Further explanation is provided by explaining that the specific 4-bit output is found by using the middle four bits and outer two bits to select a row and column from which the output is selected[4]. Knowledge of the inner workings of the S-box is not necessary for the completion of the assignment but may help in creativity and brainstorming ideas for the second, more open-ended assignment.

A simple hexadecimal string is used for the key, and helper functions are provided for translating hexadecimal strings to bit string representations and vice versa. The inspiration for using that format of strings for keys in place of a crypto key-generator library came from an article that likewise wished to reduce complexity involving the key and focus more on the algorithm[10].

We created separate functions that the students will have to correctly program in order for the input plaintext to properly convert to ciphertext and decrypt back to the original plaintext. The student will need to properly code out all functions or portions of functions marked with a "TODO" comment within the assignment. One of the first such functions encompasses the left shifting functions for shifting "bits" as seen in Figure 2, though this will be emulated through string manipulation.

More examples of the base functions that students will have to correctly implement in order to do other parts of the program that rely on them are the permutation function and the XOR function as seen in Figure 3.

Since the permutation arrays are predefined according to the DES standard, the student will have to fill out the function that takes an array and properly permutes the string/bit values according to the respective array.

```cpp
/* Shift each character in the input string one position to the left.
 *  The first character will "wrap around" and end up as the last character. */
string leftShiftOne(string in) {
    const int lenHalf = 28;
    string shifted = "";

    for (size_t i = 1; i < lenHalf; i++) {
        shifted += in[i];
    }
    shifted += in[0];
    return shifted;
}
/* Shift each character in the input string two positions to the left.
 *  The first and second characters will "wrap around" and end up as the
 *  2nd to last and last characters, respectively. */
string leftShiftTwo(string in) {
    const int lenHalf = 28;
    string shifted = "";

    for (size_t i = 2; i < lenHalf; i++) {
        shifted += in[i];
    }
    shifted += in[0];
    shifted += in[1];
    return shifted;
}
```

Figure 2: Left Shift Functions

```cpp
// Compute and return the XOR of the two input strings
string _xor(string left, string right) {
    string computed = "";

    for (size_t i = 0; i < left.size(); i++) {
        if (left[i] == right[i])
            computed += "0";
        else
            computed += "1";
    }

    return computed;
}
/* Take an input string 's' and an array 'arr' of size n.
 *  Move the character in the string's (arr[k]-1)th index to the kth spot.
 *  Example: if s = "hello" and arr = [2,5,4,1,3], then return "eolhl" */
string permute(string k, int* arr, int n) {
    // This function is short but make sure it is right as most of the DES steps will rely on this
    string perm = "";
    for (int i = 0; i < n; i++) {
        perm += k[arr[i] - 1];
    }
    return perm;
}
```

Figure 3: Permutation and XOR Functions

Figure 4 shows the round key generation portion of the code. Both left shift functions and the permute function will have to be correctly implemented to generate the sixteen round keys required for the encryption process.

```
/* Vectors for holding the 16 sub keys to be used at each of the 16 rounds of DES
*  subKeysBin has the round keys in binary, subKeysHex has the round keys in hexadecimal */
vector<string> subKeysBin;
vector<string> subKeysHex;
for (int i = 0; i < NUM_ROUNDS; i++) {
    // Shifting
    if (shift_table[i] == 1) {
        left  = leftShiftOne(left);
        right = leftShiftOne(right);
    } else {
        left  = leftShiftTwo(left);
        right = leftShiftTwo(right);
    }

    /* Combine our shifted left and right key halves, and run it through
    *  the permute function with the keyComp array to compress it down to 48 bits */
    // Combine
    string combine = left + right;
    // Key Compression
    string roundKey = permute(combine, keyComp, 48);

    /* Add the returned binary round key to our binary keys vector. Also convert it
    *  to hex and add it to the hex keys vector */
    subKeysBin.push_back(roundKey);
    subKeysHex.push_back(bin2hex(roundKey));
}

/* NOTE: At this point, you should have sixteen 48-bit keys, one for each round of DES */
```

Figure 4: Generation of sixteen round keys

3.3 Assignment 2 - AES

We were inspired by the NIST competition from which AES was chosen, to create this open-ended assignment. A new, more secure symmetric encryption algorithm was needed to replace DES. After NIST analyzed and received comments for the Round 1 entries, they selected the top five from the original fifteen[7]. Those top five were: the eventual winner Rijndael, Serpent, Twofish, RC6, and MARS. The grading criteria involved cryptanalysis, intellectual property, and crosscutting analyses[7], though the assignment we created is much simpler. As these algorithms were created by expert cryptographers who had several months to research, design, and implement them, we aimed to narrow the programming topic to a common, necessary element among them; they all created their own block cipher. The top five finalists in round two of the competition all used one of two block cipher models: Substitution-Permutation (S-P) Network/Cipher or Feistel Network/Cipher.

We wanted to have this assignment be something of a similar competition, though on a smaller scale and more for the purpose of allowing the students an opportunity to create something of their own, following one of the general paradigms of block ciphers. This allows a student to not spend so much time

on working with external libraries and interfaces and to focus more on conceiving and implementing an idea of their own choosing. A guide is provided along with the assignment and references the two main block cipher models as well as a more modern, popular model known as ARX, which stands for add-rotate-XOR, the only three operations such an algorithm is allowed to use[2]. The actual handout has code examples of very simplistic functions that do not fulfill desirable cryptography qualities but show how the initial formatting can be approached. For the assignment, students will first choose their preferred model. Students will then design and create the following functions: substitution, permutation, round-key generation. Grading for a student is based on the average number of bits of an input that are flipped by the student's functions.

The desirable cryptographic qualities are based on Claude Shannon's concepts of confusion and diffusion. Confusion obscures the relation between the ciphertext and key, and diffusion obscures the relation between the ciphertext and plaintext[9]. Effectively, we want the students to understand that the goal is to achieve a function or functions in which changing one bit of the key drastically changes the ciphertext, and similarly, changing one bit of the plaintext yields a drastically different ciphertext. From a technical point of view, the gold standard of diffusion, means that the change of a single bit of plaintext will statistically change half of the bits in the cipher text[9]. Guidance can be provided and independent research is encouraged. The assignment is purposely flexible to entail more or less complexity and work, and ultimately allow maximum student input.

4 Results

The DES assignment is available for viewing online. There are two listings: the blank assignment code, with several empty functions for students to fill out[3] and a completed assignment, as one possible example for a finished, working DES solution[11]. The created handout of the AES assignment with examples of simple implementations of block cipher components can likewise be found online for viewing or for download as a pdf[1].

5 Conclusion

There is much to learn about the world of encryption and many ways to teach it. We focused our study more on the implementations of encryption algorithms and less on the related aspects such as strength from increased key-lengths and APIs that utilize encryption. The goal we set out to achieve was the creation of assignments that teach students about symmetric encryption models and paradigms in a manner that involves more of the students' participation. We achieved this goal by creating two assignments that have students code an existing, simpler encryption and then allow them to create their own. Something

to note, these assignments provide the key to be used as the shared secret key, but in practical applications there needs to be a way to distribute and/or share that secret key. The Diffie-Helman Key Exchange is a good, possible next assignment to learn how two parties would acquire that key securely.

Another major aspect of security naturally moves toward asymmetric encryption and RSA. Secure communication is faster through symmetric encryption and one of the numerous, critical purposes that RSA fulfills is the exchange of shared secret keys. As such, it is an important and prevalent concept in understanding network security.

References

[1] AES homework with sample functions in C++. `https://jneilan.github.io/snr/aes_hw.html`.

[2] J. Aumasson, G. Leurent, W. Meier, F. Mendel, N. Mouha, Phan R. C. W., Y. Sasaki, and P. Susil. Cryptographictuple cryptanalysis of arx with application to blake and skein, 2011. `https://www.aumasson.jp/data/papers/ALMMMPSS11.pdf`.

[3] Blank homework for DES in C++. `https://jneilan.github.io/snr/des_hw.html`.

[4] J. A. Buchmann. *Introduction to cryptography*. Springer Publishing, 2001. ISBN 978-0-387-95034-1.

[5] Joan Daemen and Vincent Rijmen. The Rijndael block cipher, 1999. `https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf`.

[6] NIST. U.S. Department of Commerce Data encryption standard (DES). *F.I.P.S. PUB*, 1999.

[7] NIST. Cryptographic standards and guidelines, Dec 2016. `https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development`.

[8] B. Preneel, V. Rijmen, and A. Bosselaers. Algorithm alley, Dec 1998. `https://www.drdobbs.com/algorithm-alley/184410756`.

[9] W. Stallings. *Cryptography and Network Security (6th ed.)*. Prentic Hall, 2014. ISBN 978-0133354690.

[10] S. Upadhyay. Data encryption standard (DES) | Set 1, (2020, April 3). `https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/`.

[11] Working solution for DES in C++. `https://jneilan.github.io/snr/des_solution.html`.

How to Build and Use a Penetration Testing Environment of Virtual Machines*

Conference Tutorial

Mohamed Lotfy, Ph.D.
Utah Valley University
Orem, UT 84058
MohamedL@uvu.edu

Teaching offensive security (penetration testing/ethical hacking) is becoming a standard practice in computer science, information systems, information technology and cybersecurity programs [3, 1, 6]. Penetration testing/ethical hacking allow students to perform a sequence of different phases to gain the needed cybersecurity knowledge and skills using current tools. Through a hands-on approach, penetration testing/ethical hacking courses allow students to develop offensive cybersecurity competency enabling them later to build layered defenses that hardens the systems to penetration. Teaching penetration testing requires an attacking host that is used to perform the different phases of penetration testing on vulnerable hosts. Using an encapsulated virtual environment where the different attacks on vulnerable hosts can be conducted, reduces the risk to institutional networks and systems.

Tutorial Description

In this tutorial I will provide a hands-on working example of how to create a penetration testing environment using VMware Workstation Pro and Oracle VM VirtualBox on laptops or PCs[5, 4]. The virtual environment will include an Offensive Security Kali Linux VM, a Ubuntu Linux VM, a Metasploitable Linux VM by Rapid7, and a customized Windows XP VM.

In the tutorial I will illustrate and explain the following:

1. Offensive tools on Kali Linux
2. Port and operating system scanning using Nmap
3. Vulnerability assessment

4. Exploitation using Metasploit Meterpreter
5. How to use the Social Engineering Toolkit

Tutorial program

Step-by-step implementation and testing of a penetration testing virtual environment using multiple hosts.

Expected outcomes

Attendees will exit the tutorial with a working VMware or VirtualBox environment and learn how to perform some of the phases of penetration testing using a Kali Linux attack host.

Target audience

Any faculty who desires to incorporate a virtual environment and use it in a penetration testing/ethical hacking course.

Prerequisites

Attendees should be familiar with Linux, networking, and some programming knowledge (Java, C++, Python, etc.). It is highly recommended that attendees bring their own laptops with VMware or VirtualBox and a Kali Linux VM installed [2].

References

[1] Regina Hartley, Dawn Medlin, and Zach Houlik. Ethical hacking: Educating future cybersecurity professionals. In *Proceedings of the EDSIG Conference ISSN*, volume 2473, page 3857, 2017.

[2] KALI. Kali Linux virtual machines, 2021.

[3] Lionel Mew. The information security undergraduate curriculum: Evolution of a small program. In *Proceedings of the EDSIG Conference ISSN*, volume 2473, page 3857, 2016.

[4] ORACLE. ORACLE VM VirtualBox, 2021.

[5] VMware. VMware Workstation Pro, 2021.

[6] Yang Wang, Margaret McCoey, and Qian Hu. Developing an undergraduate course curriculum for ethical hacking. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 330–335, 2020.

Mathematical Foundations of Computer Science*

Conference Tutorial

Pradip Peter Dey and Hassan Badkoobehi
Department of Engineering and Technology
National University
San Diego, CA 92123, USA
{pdey, hbadkub}@nu.edu

Abstract

The foundational aspects of computer science are usually explained with mathematical models such as Finite Automata (FA), Pushdown Automata (PDA) and Turing Machines (TMs) because understanding the scope and limitations of computation are inherently embedded in an overlapping area of mathematics, logic and cognition. This area is one of the most challenging fields in computer science. Students tend to engage, and participate actively when the basic ideas are introduced in a tutorial with multiple perspectives in multiple representations including transition graphs, tabular forms and coded strings. Mathematical proofs are provided in a gentle introductory session in order to help beginners.

Tutorial Description

The central ideas of computation are defined in an area commonly known as the theory of computation or automata theory that comes from an interesting overlap between mathematics, logic and cognition. Most learners will benefit from a tutorial at their initial stages of learning. In order to have good interactions with learners, multiple perspectives and multiple representations are used for initial discussion of mathematical ideas[1, 2, 3, 4, 5, 6]. Mathematical models including FA, PDA, and TMs are explained with examples; their processing styles and computing powers are compared with references to languages they accept. The tutorial starts with explaining how FA can process regular languages defined by regular expressions. That PDA can accept

Context Free Languages (CFLs) is explained next. Finally, TMs are explained with non-CFLs. Undecidable problems are explained with examples such as integral polynomials[6]. Some supplemental materials from the following site are used in this tutorial: `http://www.asethome.org/mathfoundations/`

Expected outcomes

Attendees will exit the tutorial with a good understanding of mathematical foundations of computer science. Attendees will have free access to the supplemental materials at the following site:
`http://www.asethome.org/mathfoundations/`

Target audience

Any faculty who desires to teach mathematical models of computation.

Prerequisites

None. Everybody is welcome to this tutorial. Intuitive explanations are presented in this tutorial with the assumption that the attendees do not have any knowledge of the theory of computation, and they may be interested in questions such as: What is computable? What is not computable?

References

[1] S. Ainsworth. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16:183–198, 2006.

[2] A. Alkhatla and J. Kalita. Intelligent tutoring systems: A comprehensive historical survey with recent developments. *International Journal of Computer Applications*, 2019.

[3] D. Cohen. *Introduction to Computer Theory.* John Wiley & Sons, 2nd edition, 1996.

[4] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Pearson Education, 2nd edition, 2007.

[5] S. Rodger and T. Finley. *JFLAP: An interactive formal languages and automata package.* Jones & Bartlet Learning, 2006.

[6] M. Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 3rd edition, 2012.

Ethical Considerations in Undergraduate Online Computer Science Curriculums*

Lightning Talk

Bhaskar Sinha, Pradip Peter Dey,
and Mohammad Amin
National University
San Diego, California, 92123, USA
{bsinha, pdey, mamin}@nu.edu

The coronavirus (COVID-19) epidemic environment has made academic institutions evaluate alternatives to traditional onsite classroom education. One of the options being studied is a strong and effective online system with virtual classrooms using state-of-the-art Learning Management Systems (LMS). In many Science, Technology, Engineering, and Math (STEM) scenarios, especially in Computer Science (CS) curriculums, these online systems perform as good as, or better than, the traditional onsite systems. Going forward, new challenges and opportunities with this new educational paradigm must be critically examined, and CS educators must strive to learn how to effectively teach in this online setting. This research focusses on undergraduate CS classes where most of the content is delivered virtually with the aid of online technologies. As more students enroll in virtual courses, questions, and concerns about ethical teaching practices in digital classrooms continue to surface. Handling of due process, freedom of expression, diversity among students, individual rights, etc. are among the many critical issues that need to be evaluated and carefully integrated in the content delivery and evaluation processes. Teachers are ethically accountable to serve the learning needs of all learners, and to do this they must recognize, understand, and appreciate the cultural backgrounds, values, beliefs, world views, and wide-ranging experiences that students bring to the class. Instructors must understand the motivations of these culturally diverse group of students to be able to connect with them, to motivate them, and to create a positive learning environment for the learners. CS education is important because it teaches logical and evidence-based critical thinking skills and develops a desire for innovation. Beyond the benefit of learning STEM, CS

curriculums help learners in the problem-solving and exploratory learning that leads to success across a variety future endeavors across disciplines. Access to CS education is a social justice issue that goes far beyond the latest trends. It is becoming increasingly apparent that we are at a crucial point when those who effectively apply CS concepts can improve the lives of many and contribute to the acceleration of profound societal advancements. As educators, teachers in academia have the responsibility to leverage the power of STEM, including CS, to help address disparities in the immediate communities and beyond, which would hopefully result in significant and positive progress. This research argues that CS undergraduate education must serve as a context for moral development by expanding student argumentation and discourse to include the moral and ethical consequences of decision making. This study contains some current thoughts in ethics education in CS curriculums including intellectual engagement versus emotional engagement, the curricular structures, the nature of engineering faculty, and the engineering of the topic of ethics.

Based on the above background, this research analyzes some of the areas of digital privacy, intellectual property, and professional practice, in teaching CS courses in undergraduate programs and suggests some best practices to deal with these challenges. This ongoing study is narrowed to undergraduate online CS programs due to the backgrounds of the authors and their experiences in teaching in this space. It is expected that this preliminary report and the lightning talk will generate interest, questions, and suggestions from the audience, thus enhancing the scope and the quality of this effort.