The Journal of Computing Sciences in Colleges

Papers of the 32nd Annual CCSC South Central Conference

April 9, 2021
The University of Texas at Dallas
Richardson, TX

Baochuan Lu, Editor
Southwest Baptist University

Laura Baker, Regional Editor
St. Edward's University

Volume 36, Number 7 **April 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.

Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors **5**

CCSC National Partners **7**

Welcome to the 2021 CCSC South Central Conference **8**

Regional Committees — 2021 CCSC South Central Region **9**

Reviewers — 2021 CCSC South Central Conference **10**

We're All In This Together: Learning Communities for First-Year Computer Science Majors **11**
Rita Sperry, Texas A&M University - Corpus Christi

Logarithmic Rubric Scoring **20**
Jun Rao, James Palmer, Northern Arizona University

MUIPC and Intent to Change IoT Privacy Settings **27**
C. Bryan Foltz, Laura Foltz, University of Tennessee at Martin

Narcoleptic Philosophers: Message Delays on Paths with ON/OFF Nodes **39**
Timothy Nix, Stephen F. Austin State University, Riccardo Bettati, Texas A&M University

Towards a Software System for Spatio-Temporal Authorization **47**
Mustafa Al Lail, Marshal Moncivais, and Miguelangel Trevino, Texas A&M International University

Use of Predefined Computing Blocks in Algorithmic Thinking **56**
S.R. Subramanya, University of Central Oklahoma

A Bottom-Up Approach for Computer Programming Education **66**
Lasanthi N. Gamage, Webster University

Making Change: Rigorous Software Construction as Experiential Learning **76**
Michael Kart, Saint Edward's University

Techniques for Effective Teaching and Learning in the Wake of Transition to Online Classes Due to COVID-19
— Conference Tutorial **86**
Srikantia Subramanya, University of Central Oklahoma

Enrich Student Learning Experience by Building a Cybersecurity Virtual Lab with Open-Source Tools
— Conference Tutorial **88**
Jianjun Zheng, Stephen F. Austin State University

Play With Trained Hierarchical Reinforcement Learning Agents in Two Common Games
— Conference Tutorial **89**
Chengping Yuan, Mark V. Albert, Daniel McGartland, Jakob Smith, Anthony Solorio, University of North Texas

On-ramp to AI: lessons from the introductory AI course "Software Development for AI"
— Conference Tutorial **91**
Ting Xiao, Mark V. Albert, University of North Texas

Programming With the Cloud
— Conference Tutorial **93**
Laurie White, Google Cloud

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, School of Computing and Engineering, 5110 Rockhill Road, 546 Flarsheim Hall, University of Missouri - Kansas City, Kansas City, MO 64110.

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 "Crystal" Peng, Associate Editor, (816) 584-6884, crystal.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 South Central Conference

The 2021 South Central Steering Committee is very pleased to welcome everyone to our 32nd annual conference online hosted by the University of Texas at Dallas. Our conference chair and host, Shyam Karrah, has provided infrastructure and support for the virtual delivery of our conference. With generous time and effort the Steering Committee has delivered a fine program overcoming the challenges of virtual hosting and production of this year's program.

For our online 2021 conference we have eight papers, five workshops, and both student and faculty posters scheduled for the program. This year the Steering Committee chose 8 of 17 papers through a double-blind review process for a paper acceptance rate of 47%. Seventeen colleagues across the region and country served as professional reviewers and we recognize the expertise and guidance they all so thoughtfully contributed to the selection of our 2021 conference program.

The Steering Committee continues to seek colleagues to host the conference in the future and to join our community of computer science educators to enrich our curricula and provide innovative pedagogy for our students. We invite and encourage our fellow members of the South Central region to attend our Monday April 12, 2021 virtual evening business meeting. Fellow educators and colleagues are encouraged to join in our efforts to involve more of our community in the planning and execution of the conference in the future.

We extend a very warm and delightful welcome to our virtual presenters and attendees who continue to promote computer science education and camaraderi to our region. To all members of our 2021 Steering Committee, thank you again for your gracious efforts in delivering our first virtual conference during such challenging times.

Shyam Karrah
University of Texas at Dallas
Conference Chair and Host

Laura J. Baker
St. Edward's University
Papers and Program Chair

2021 CCSC South Central Conference Steering Committee

Shyam Karrah, Conference Chair ..... University of Texas at Dallas, TX
Laura Baker, Papers/Workshops Chair ..... St. Edward's University, TX
Michael Scherger, Posters Chair ..... Texas Christian University, TX
Bingyang Wei, Moderators Chair ..... Texas Christian University, TX
Eduardo Colmenares-Diaz, Publicity Chair Midwestern State University, TX
Michael Kart, Nifty Assignments Chair ..... St. Edward's University, TX
Tim McGuire, At-Large Member ..... Texas AM University, TX

Regional Board — 2021 CCSC South Central Region

Anne Marie Eubanks, Registrar ..... Stephen F. Austin State University, TX
Tina Johnson, National Board Representative .. Midwestern State University, TX
Bilal Shebaro, Treasurer ..... St. Edward's University, TX
Laura Baker, Regional Editor ..... St. Edward's University, TX
Abena Primo, Webmaster ..... Huston-Tillotson University, TX

Reviewers — 2021 CCSC South Central Conference

Baker, Laura ..... St. Edward's University, Austin, TX
Krishnaprasad, Srinivasarao .. Jacksonville State University, Jacksonville, AL
Gurney, David ..... Southeastern Louisiana University, Hammond, LA
Rouse, Kenneth ..... LeTourneau University, Longview, TX
Metrolho, Jose ..... IPCB, Portugal, Castelo Branco, CB
Scherger, Michael ..... Texas Christian University, Fort Worth, TX
Song, Hongzhi .... Texas AM University-Corpus Christi, Corpus Christi, TX
Shebaro, Bilal ..... St. Edward's University, Austin, TX
Williams, Chadd ..... Pacific University, Forest Grove, OR
Renwick, Janet ..... University of Arkansas–Fort Smith, Fort Smith, AR
Primo, Abena ..... Huston-Tillotson University, Austin, TX
Johnson, Tina ..... Midwestern State University, Wichita Falls, TX
Colmenares, Eduardo ..... Midwestern State University, Wichita Falls, TX
Rahman, Muhammad ..... Clayton State University, Morrow, GA
Edwards, Andrea ..... Xavier University of Louisiana, New Orleans, LA
Kart, Michael ..... St. Edward's University, Austin, TX
Wei, Bingyang ..... Texas Christian University, Fort Worth, TX

We're All In This Together: Learning Communities for First-Year Computer Science Majors*

Rita Sperry
Department of Undergraduate Studies
Texas A&M University - Corpus Christi
Corpus Christi, TX 78412
rita.sperry@tamucc.edu

Abstract

In Fall 2016, the learning communities program at Texas A&M University Corpus Christi, a four-year public institution in Texas, offered its first learning community for incoming Computer Science majors that linked the introductory Computer Science course with a Mathematics course and First-Year Seminar in an effort to improve student performance in first-year Computer Science courses and increase retention rates for students in the major and at the institution. Over the course of the past four years, the faculty members in the Computer Science learning community have developed an assignment sequence that integrates the first-year experience and takes advantage of the inherent benefits of the learning community model in order to promote student success and engagement. Preliminary analysis of grades in the Computer Science introductory sequence courses and retention figures after the first and second year for Computer Science learning community students appear to indicate that the creation of this learning community has had a positive impact on the first-year experience for incoming Computer Science majors.

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Learning communities (or LCs) involve the linking of "two or more courses, often around an interdisciplinary theme or problem, and enroll a common cohort of students"[6]. They are considered a high-impact educational practice[4] and can be found at a majority of the nation's colleges and universities[2]. LCs that include a first-year seminar course are designed to help student and faculty make connections with one another across course and disciplinary boundaries, with the ultimate goal of developing students' abilities to engage in deep, integrative, and lifelong learning[3].

Faculty members who teach in LCs are expected to collaborate before and during the semester to intentionally align their goals, calendars, and assignments to take full advantage of the learning community structure. Many LC experiences require students to complete integrated assignments that challenge them to leverage the knowledge and skills gleaned from all of their linked courses to tackle novel problems or issues. Beyond helping students to develop important communication, teamwork, and critical thinking skills, LCs have been shown to positively impact student success measures such as first-year GPA and retention[1].

At Texas A&M University-Corpus Christi (TAMU-CC), all full-time first-year students are required to participate in LCs during their first two semesters. Historically, these LCs have included one to three linked core curriculum courses open to all majors connected with First-Year Seminar. However, the growing number of students bringing in dual credit coursework has led to the recent development of major-specific LCs that included classes outside of the core curriculum. The Computer Science LC was the first LC at TAMU-CC to include courses that were not in the core curriculum, which set the stage for other disciplines (eg, Engineering, Music, Art) to offer their own major-specific LCs for first-year students in recent years. As of Fall 2020, nearly half of the LCs offered each semester at TAMU-CC are reserved for particular majors.

2 The Computer Science LC at TAMU-CC

The Computer Science LC was intentionally designed to support the transition of incoming first-year Computer Science majors while they took their first introductory programming class at TAMU-CC – COSC 1435: Introduction to Problem-Solving with Computers I. This class had traditionally posed challenges to incoming students and the thought was that linking it to a First-Year Seminar section designed to support first-year student success would benefit student engagement and performance. Students in this LC also registered for Trigonometry or Calculus together with other students in the LC based on their

math placement. In 2018, COSC 1100 (Skills for Computing Professionals) was added to the fall LC schedule.

In the spring semester, first-year students who pass the COSC 1435 course have the option to continue with the Computer Science LC to take COSC 1436, MATH 2305 (Discrete Mathematics), and the second required semester of First-Year Seminar. Computer Science LC students have also had the option to add major-specific sections of COMM 1311 in the fall and ENGL 1302 in the spring if desired. The Computer Science LC courses are summarized in Table 1 below, while Table 2 contains a breakdown of students in the Computer Science LC by semester since its inception in Fall 2016.

Table 1: Computer Science LC Courses, 2016-Present

Fall Computer Science LC (2016-Present)
COSC 1435: Introduction to Problem Solving with Computers I
COSC 1100: Skills for Computing Professionals (added Fall 2018)
MATH 1316: Trigonometry or MATH 2413: Calculus I
UNIV 1101: First-Year Seminar I
Optional COMM 1311 course
Spring Computer Science LC (2017-Present)
COSC 1436: Introduction to Problem Solving with Computers II
MATH 2305: Discrete Mathematics I
UNIV 1102: First-Year Seminar II
Optional ENGL 1302 course

Table 2: Computer Science LC Enrollments, 2016-Present

Semester	Enrollments
Fall 2016	47
Spring 2017	32
Fall 2017	55
Spring 2018	39
Fall 2018	35
Spring 2019	25
Fall 2019	31
Spring 2020	19

3 First-Year Seminar and the Integrated Computer Science LC Experience

The First-Year Seminar curriculum at TAMU-CC is based on an adaptive, just-in-time model that allows for the incorporation of student success topics (such as notetaking, time management, study skills, campus resources) in the context of the linked learning community courses at the particular time of need[3]. The student learning outcomes include integrative learning, lifelong learning, higher education navigation, and academic development. The course meets twice per week for fifty minutes and is highly interactive in nature, with a heavily reliance on in-class discussion and participation.

Daily activities and topics in First-Year Seminar can include, but are in no way limited to, the following: group tasks or discussions that allow students to practice and grapple with concepts from the linked courses; exam reviews; visits from campus resources; course registration and academic advising; habits of mind and growth mindset; personal values, goals, and strengths; career exploration and professional development; and metacognitive reflections on learning. Faculty members who teach First-Year Seminar do not necessarily have expertise in the linked courses, but the program typically attempts to match First-Year Seminar faculty based on their background and preference for working with particular disciplines.

In the Computer Science LC, the First-Year Seminar course necessarily supports the specific demands of the linked Computer Science and Mathematics courses, while at the same time incorporating timely discussions about first-year student resources, strategies, and requirements. All of the faculty members in the Computer Science LC for First-Year Seminar, Computer Science, and Mathematics meet prior to the start of each semester to align due dates and deadlines for exams and to established shared learning outcomes for their students. The First-Year Seminar professor attends the COSC 1435/6 class with the Computer Science LC students throughout the semester and regularly communicates with instructors for the other linked courses to discuss upcoming assignments, provide feedback on student concerns, and discuss students who might be struggling. Frequently the faculty members will work together to develop a plan for responding to individual student issues.

Over the years, the Computer Science LC faculty members have developed an assignment sequence that includes two major integrative projects that are facilitated within the First-Year Seminar course and are designed to help students to make connections between their LC courses, practice necessary teamwork and communication skills, and enhance their ability to identify the relevance of their coursework to their future careers. During the fall semester, students work together in teams to design and code a simple text-based game.

The spring semester includes an experience affectionately entitled the First-Year Capstone Project in which the students are challenged to implement a project for a real-world client.

3.1 Fall Team Coding Project

About a third of the way into the fall semester, the Computer Science LC students are placed into teams in First-Year Seminar and are challenged to create a pitch for a text-based game for an imaginary potential client who is interested in the first-year student experience. Teams are encouraged to be creative with their storylines, with the only major caveat being that the game must relate somehow to real problems faced by first-year students at TAMU-CC. As they learn about the problem-solving process and the design of algorithms in the linked COSC 1435 course, they work together in First-Year Seminar to create flowcharts and pseudocode for particular portions of their game. Once students begin to learn more about coding in C++, they then implement their team's game design to include, at a bare minimum, basic decision and loop structures, input and output from files, functions, and arrays/vectors. Teams are highly encouraged to go beyond the minimum – and many choose to do so – and, over the years, the student teams have figured out creative ways to incorporate color, images, animation, and features like high score tracking and save files into their designs. Teams typically have the opportunity to celebrate and showcase their final products to the campus community at First-Year Symposium, a first-year student poster session required of all learning community students at TAMU-CC.

The fall coding project allows students to get their first real experience working together as a team to complete a complex coding challenge, an activity that they will necessarily engage in throughout their undergraduate Computer Science experience at TAMU-CC but, perhaps more importantly, throughout their careers if they choose to stay in Computer Science. Reflection is built into the assignment sequence for the project, and discussions about the importance and challenges inherent in collaboration are a common topic of discussion in the First-Year Seminar course throughout the semester. The project also allows students to apply the material from their linked COSC 1435 course in a way that is substantively different from their traditional lab or homework assignments. It is not uncommon for students to express excitement at how a concept they learned in the previous COSC 1435 lesson could be immediately incorporated into their team's game design. The level of student engagement each fall semester in the team coding project consistently surprises and excites the Computer Science LC faculty teaching team, and it serves as an exemplar of all of the features necessary for a high-impact learning experience[4].

3.2 Spring First-Year Capstone Project

During the spring semester, Computer Science LC students are again placed into faculty-selected teams to complete the First-Year Capstone Project. Intended to simulate the senior capstone experience for Computer Science majors at TAMU-CC, teams are challenged to implement a program that solves a problem for a real-world client. Because the linked COSC 1436 course focuses on object-oriented programming, the one major requirement for the solution – aside from needing to be written in C++ – is that it must include at least one class. The assignment sequence mirrors the senior capstone stages of requirements collection and analysis, design, implementation, and testing. Accordingly, student teams are tasked with reaching out and meeting with their client multiple times to define and clarify requirements and demonstrate drafts of their solutions for feedback. Major deliverables for the project include requirements and design documents, rough drafts and final code submissions, and a final formal presentation that includes a live demonstration of the solution.

This project is meant to introduce students to the software development life cycle in a supportive and "safe" environment. As such, the "clients" for their projects have always been faculty members or peer mentors in the Computer Science LC whose problems related to their role in the learning community. The ENGL 1302 instructor, for example, has challenged his student groups to create a "better" APA/MLA citation generator, while the Discrete Mathematics instructor asked his assigned teams to create a truth table builder. This is yet another example of how students in the Computer Science LC are challenged to integrate the concepts they are learning across the curriculum.

4 Results

Table 3 below includes the one-year retention rates of first-year students at TAMU-CC from 2012 to 2019. The first row lists the overall retention figure for all first-year students, while the second row filters the overall list to first-year Computer Science (CS) majors. The third row refines the group even further to first-year Computer Science majors who participated in the Computer Science learning community experience during their first semester.

Table 3: First-Year Student One-Year Retention, 2012-2019

	2012	2013	2014	2015	2016	2017	2018	2019
Overall	61.6	58.8	62.7	60.9	57.3	57.8	59.8	57.8
CS Majors	61.8	60.9	70.7	70.4	67.5	67.1	69.5	76.0
CS LC					74.5	72.9	74.5	75.0

Table 4 contains the two-year retention rates of first-year students at TAMU-CC from 2012 to 2018. Again, the first row represents all first-year students, the second row represents all first-year Computer Science students, and the third row only includes students who participated in the Computer Science LC during their first fall semester.

Table 4: First-Year Student Two-Year Retention, 2012-2018

	2012	2013	2014	2015	2016	2017	2018
Overall	48.2	45.1	49.1	48.0	45.2	43.7	47.8
CS Majors	50.9	56.3	56.0	56.8	53.8	50.6	61.0
CS LC					61.7	55.9	72.3

Both of the above tables appear to indicate that students in the Computer Science learning community are more likely to return for their second and third years than the general student population. While the gap between the overall one-year retention rates of first-year students and Computer Science LCs is rather large – a 17.2 point difference in the 2019 cohort – the two-year retention rate differences are even more striking. There is likely some selection bias playing a role in this difference because students in the Computer Science LC likely have higher math placements than the average first-year student due to the prerequisites for entry into the COSC 1435 and linked MATH courses.

Beyond retention, the primary goal of the Computer Science LC is to improve performance in the linked Computer Science introductory sequence courses. Tables 5, 6, and 7 below include the C or better rates for COSC 1435, 1436, and 2437, the first three course that students take as Computer Science majors. The first two courses are taken as part of the LC experience in the first year, while COSC 2437 is typically taken during the student's second fall semester. Table 7 only includes students who took the Computer Science LC during both semesters of their first year.

Table 5: COSC 1435 C or Better Rate, Fall Semesters, 2012-2019

	2012	2013	2014	2015	2016	2017	2018	2019
Overall	65.4	55.9	40.5	61.6	75.8	68.1	59.6	55.6
CS Majors	69.3	57.7	38.4	68.0	81.2	67.8	62.5	67.2
CS LC					87.2	76.4	68.6	67.7

Thus, students who participate in the Computer Science LC are not only more likely to be retained in to their second and third fall semesters, their performance in their introductory Computer Science course sequence (1435,

Table 6: COSC 1436 C or Better Rate, Spring Semesters, 2013-2020

	2013	2014	2015	2016	2017	2018	2019	2020
Overall	48.8	52.9	51.7	57.4	69.4	40.0	72.6	59.3
CS Majors	49.3	53.5	52.2	59.0	67.9	43.1	75.0	59.2
CS LC					87.5	53.9	96.0	63.2

Table 7: COSC 2437 C or Better Rate, Fall Semesters, 2014-2019

	2014	2015	2016	2017	2018	2019
Overall	71.7	60.0	65.5	88.4	77.8	57.9
CS LC				92.6	88.9	81.8

1436, and 2437) is higher than students who take those same classes outside of the learning community experience. Perhaps the most significant finding is the performance of Computer Science LC students in the third programming course, COSC 2437, since that course is taken after completing the learning community experience.

5 Discussion

After its fourth year of implementation, the results of the Computer Science LC experiment at TAMU-CC are promising. Students in the LC earn higher grades in the linked Computer Science classes (COSC 1435 and COSC 1436) than other students in the same classes who are not in the LC, and they go on to perform better in the next class they take during their second fall semester (COSC 2437). In addition, the one- and two-year retention rates for first-year Computer Science majors in the LC are higher than the general population.

Much more research is needed to explore the qualitative nature of the benefits of learning communities on first-year student success, particularly in Computer Science. Anecdotally, faculty members who teach the linked Computer Science and Mathematics courses indicate that the LC students are much more engaged in the experience, have greater in-class attendance and participation, tend to ask more questions, and identify peer and study groups more easily than their non-LC colleagues. The benefits of learning communities identified by others as hallmarks of a high-impact educational practice[5] seem to be manifesting themselves in the Computer Science LC experience at TAMU-CC.

References

[1] M. S. Andrade. Learning communities: Examining positive outcomes. *Journal of College Student Retention*, 9(1):1–20, 2008.

[2] B. O. Barefoot, B. Q. Griffin, and A. K. Koch. *Enhancing student success and retention throughout undergraduate education.* John N. Gardner Institute for Excellence in Undergraduate Education, Brevard, North Carolina, 2012.

[3] L. Chism Schmidt and J. Graziano. *Building synergy for high-impact educational initiatives: First-year seminars and learning communities.* National Resource Center for the First-Year Experience and Students in Transition, Columbia, South Carolina, 2016.

[4] G. D. Kuh. *High-impact educational practices: What they are, who has access to them, and why they matter.* Association of American Colleges and Universities, Washington, DC, 2008.

[5] G. D. Kuh and K. O'Donnell. *Ensuring quality and taking high-impact practices to scale.* Association of American Colleges and Universities, Washington, DC, 2013.

[6] B. Smith, J. MacGregor, R. Matthews, and F. Gabelnick. *Learning communities: Reforming undergraduate education.* Jossey-Bass, San Francisco, California, 2004.

Logarithmic Rubric Scoring*

Jun Rao[1], James Palmer[1]
[1]School of Informatics, Computing, and Cyber Systems
Northern Arizona University
Flagstaff, AZ 86001
jr2339@nau.edu, James.Palmer@nau.edu

Abstract

Grading rubrics communicate convenient explicitly articulated performance expectations to students and provide a consistent methodology for assessment, but using linearly weighted rubrics can lead to poor grade differentiation among high performing students and as a percentile score can often poorly map to instructor academic achievement expectations. In this work, we describe a logarithmically weighted rubric for computer science projects and analyze its effect on grading through modeling and surveys.

1 Introduction

In 1913, E. Finkelstein expressed concern about conventional grading practices as part of a statistical monograph on grading. He begins his work by asserting "the percentage system with 100 for a maximum and 60 or 70 as a 'pass mark' is in all probability not the best system" [8]. Some 80 years later, Mark Durm similarly raised concerns about the current grading system asking the question "Is our current grading system still uncalibrated?" [7]. Cross and Fray pointed out that while there is no consensus on the most appropriate grading system; there is, however, quite a bit of agreement about what is wrong with the current system [6]. Unfortunately, as stated by Cizek, even as "grades continue to be relied upon to communicate important information about academic performance... they probably don't" [5].

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

For these reasons, educational researchers and theorists have been highly critical of traditional grading practices, and more than that, some researchers have proposed abolishing traditional grading practices [11, 12, 1].

Because the sole purpose of a grade is to accurately communicate to others the level of academic achievement that a student has obtained [14], do the assessment procedures and assignments of grades accurately reflect and communicate the academic achievement of students? The standard grading system may actually be a barrier as it requires unnatural mappings to a zero to one hundred scale. At best this can be cumbersome and it worst it doesn't deliver the truth about academic achievement [4].

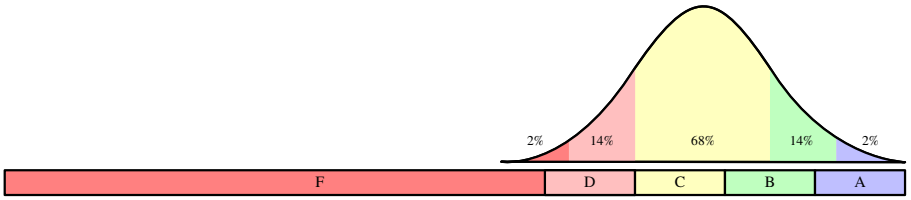


Figure 1: The most common grading scale, arguably, doesn't strongly align with the student performance we expect.

Consider what many believe to be a mathematical ideal where grades are aligned with a normal distribution such that most grades fall in the 'C' range and roughly 85% of grades are passing. Yet, as illustrated in Figure 1, the actual scale we use for assigning grades awards 70% of the points to the performance that we might expect from 2% of the student population. This problem is particularly acute for many standard grading rubrics.

Though the word 'rubric' comes from the Latin 'marks in red' (Finson and Ormsbee, 1998) [9], today, a rubric is understood to be a coherent set of grading criteria based on descriptions of performance. Rubrics are essentially a scoring tool that indicates 'what counts' [2, 13] but do so using categories of performance and not explicitly a traditional percentile scoring system.

While there are strong reasons to upend traditional grading systems altogether, there are problems with that too. Students, in general, don't like inconsistency. Non-traditional grading systems are more often the target of grade appeals and when different classes use dissimilar grading systems this can be a point of frustration for students who feel that benchmarks and standards are being unfairly changed.

Rather than change student attitudes or completely change how we do grading, we propose adopting a rubric grading scale with performance levels aligned with the performance we expect but weighted with points such that the final score uses a traditional 0 to 100 scale. We suggest this is often done

informally but in this work we formally analyze this approach and describe a logarithmic relationship between rubric performance levels and the percentage grade scale.

2 Rubric Design

Many institutions provide guides and workshops with best practices and design tips for rubrics. The mechanics of a rubric are straightforward - a grid is formed from a pairing of dimensions with performance levels.

Dimensions or criterion used in a rubric are typically enumerated on the left column of the rubric and broadly categorize the activities instructors are looking for in the assessed work. Three to five criteria are the most common [3]. The example rubric in Figure 2 uses three common computer science dimensions: correctness, design, and style, weighted at 50%, 25%, and 25% respectively. *Correctness* describes the degree to which the code meets technical specifications. *Design* describes the degree to which the code is efficient and well organized. And *style* represents the degree to which the code is readable including variable naming, commenting, and indentation. In addition to having lower weight, design and style are "parameritized" by correctness. That is, design and style must support the correctness of the code - these dimensions are not independent. It would be unlikely, for example, to get high scores in design and style and a low score in correctness.

Performance levels provide broad categories of student expectations, listed at the top of the rubric, and when paired with a criterion describe an instantiation of the performance level with the criterion. Four to five performance levels are the most common [3] but four to six levels are typically recommended [10]. An odd number of performance levels can create an implicit catch-all category and limit a more precise judgement of student performance.

Technical objectives, while not a feature of the rubric, support the dimensions and performance levels by articulating the specific features that should be implemented in the project. This allows the rubric to speak more generally about project expectations and emphasizes specific technical requirements in a check list attached to the assignment.

3 Logarithmic Rubric Weighting

A logarithmic rubric weighting is one which has exponentially increasing penalties. Specifically, we considered the equation ik^p, where p is the performance level, i is the initial penalty and k is the growth constant. For $c = 5$ and k chosen such that $5k^5 = 100$, we find the score progression $(100 - penalty)$ is $\{100, 95, 90.9, 83.4, 69.8, 45.1, 0\}$, charted in Figure 3. The actual progression

	No Effort	Poor Effort	Fair Effort	Good Effort	Very Good Effort	Extremely Good Effort
Correctness	Missing or trivial effort; program did meet technical objectives or simply did not work and minimal evidence was available that an effort was made.	Major issues were present; significant deficiencies related to meeting technical objectives.	Ample room for improvement; program worked in part but not all technical objectives were met.	Some areas for improvement; most technical objectives achieved but some areas for improvement or corner cases were missed.	Minor areas for improvement; all technical objectives were achieved with perhaps minor areas for improvement or minor corner cases not addressed.	Strong attention to detail; all technical objectives were achieved with strong attention to detail on corner cases.
50%	**0%**	**45%**	**70%**	**85%**	**95%**	**100%**
Design	Missing or trivial effort; lacked an effort toward any meaningful organization.	Major issues were present; serious problems in code organization, design or efficiency.	Ample room for improvement; design was functional but there is significant space for improvement.	Some areas for improvement; design is acceptable but could be improved.	Minor areas for improvement; very good design with only minor areas for improvement.	Strong attention to detail with especial attention to modularity, code reuse, and architecture.
25%	**0%**	**45%**	**70%**	**85%**	**95%**	**100%**
Style	Missing or trivial effort; code had incoherent readability and organization.	Major issues were present or lacked a serious effort; Poor variable naming, indentation, and readability.	Ample room for improvement; code was basically readable but there is significant room for improvement in variable naming, indentation, or other stylistic elements.	Some areas for improvement; code is fairly readable but there are minor inconsistencies or missing comments.	Minor areas for improvement; very good style with only minor inconsistency or room for improvement in commenting.	Strong attention to detail with excellent commenting and attention to detail in style and thinking about audiences and communication.
25%	**0%**	**45%**	**70%**	**85%**	**95%**	**100%**

Figure 2: A generic computer science project rubric with associated dimensions, performance levels, and weights.

of scores we use in our rubric is rounded to the nearest multiple of five as not to indicate to the student a level of precision that is not present.

This scoring system has some interesting properties. Four of the six performance levels are associated with passing scores (66.7%) and only two of the six levels are associated with failing scores (33.3%). This is almost the reverse allocation compared to the traditional percentile scale where 59-69% of points represent unacceptable scores. Another effect is that performance level descriptions don't over represent unacceptable performance and instead spend more time differentiating acceptable levels of performance.

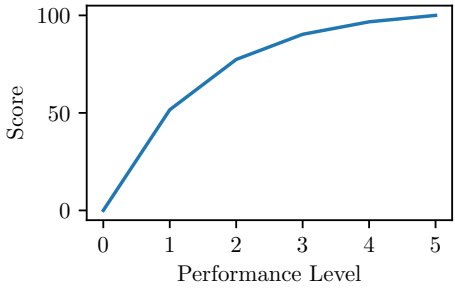


Figure 3: Scores and performance levels reveal a logarithmic relationship.

This scale is applied for each dimension of the rubric. The example rubric in Figure 2 illustrates three dimensions but we can visualize how the rubric would be applied to 1, 2, 3, or even a higher number of dimensions. Figure 4 demonstrates, as one might expect, that as the number of dimensions increases the density at the upper end of the scale increases faster than the density at the lower end of the scale. If average performance for a class falls between "Fair Effort" and "Good Effort" and there is a normal distribution of grades we can expect a curve that looks like Figure 1 without additional mathematical "post processing".

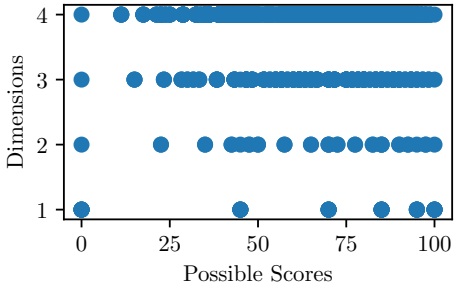


Figure 4: As the score weights are applied to progressively more dimensions the density at the upper end of the scale increases faster than the lower end.

Another positive effect of this system is that it gives some of the highest performing students more feedback and grade differentiation. This allows instructors to set a high bar in the rubric for "Very Good Effort" and "Extemely Good Effort" while not making an 'A' or a 'B' impossible.

4 Deployment and Student Feedback

Variations of this rubric and logarithmic scoring has been deployed in several classes. Most recently, this rubric was used in an upper division computer science elective class with six large open-ended project-based assignments. BbLearn was used as the learning management system for the course and all projects were collected as BbLearn assignments. The BbLearn rubric tool was used to design and implement the rubric and weightings as described in this paper. The same rubric was used for each project but with differing technical objectives.

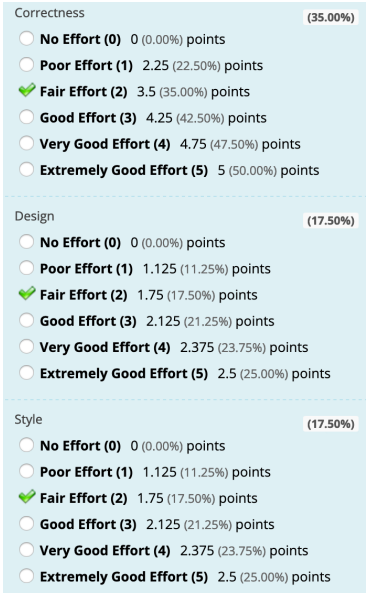


Figure 5: The rubric scoring user interface used in BbLearn.

Assignments are graded based on input from automated tests constructed for the class and linting tools. In the past a git "linting" tool has also been used to assess version control usage. These inputs along with an inspection of the code and accompanying documentation inform the performance level assignment for each dimension and each student.

Feedback was collected in the form of surveys with 43 students responding. A Likert scale from 1 to 10 was used to assess student satisfaction with mapping of performance levels to scores for projects. Students reported high satisfaction with the rubric, specifically reporting a 9.5 satisfaction rating.

5 Conclusion

We suspect that logarithmic scoring has been used informally in rubrics, but has not previously been studied explicitly. This work describes the motivation for such a grading scale, its implementation, and its deployment in the classroom. Students who used the scale have indicated that they were satisfied with the scale and analysis of synthetic models and real scores suggest it maps student achievement as desired.

While this work included a small student satisfaction survey, a larger study is needed to better understand how rubric scoring impacts student learning.

References

[1] James D Allen. Grades as valid measures of academic achievement of classroom learning. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, 78(5):218–223, 2005.

[2] Heidi Goodrich Andrade. Understanding rubrics. *Educational Leadership*, 54(4):14–17, 1997.

[3] Susan M. Brookhart. Appropriate criteria: Key to effective rubrics. *Frontiers in Education*, 3:22, 2018.

[4] Enwefa Chiekem. Grading practice as valid measures of academic achievement of secondary schools students for national development. *Journal of Education and Practice*, 6(26):24–28, 2015.

[5] Gregory J Cizek. Grades: The final frontier in assessment reform. *Nassp Bulletin*, 80(584):103–110, 1996.

[6] Lawrence H Cross and Robert B Frary. Hodgepodge grading: Endorsed by students and teachers alike. *Applied Measurement in Education*, 12(1):53–72, 1999.

[7] Mark W Durm. An A is not an A is not an A: A history of grading. In *The educational forum*, volume 57, pages 294–297. Taylor & Francis, 1993.

[8] Isidor Edward Finkelstein. *The Marking System in Theory and Practice.* Baltimore: Warwick & York, 1913.

[9] Kevin D Finson and Christine K Ormsbee. Rubrics and their use in inclusive science. *Intervention in School and Clinic*, 34(2):79–88, 1998.

[10] Centre for Enhanced Teaching and Learning. Grading rubrics: Set expectations, make feedback delivery more efficient. `https://unbtls.ca/teachingtips/gradingrubrics.html`.

[11] A Kohn. Grading is degrading. *Education Digest*, 65(1):59–64, 1999.

[12] Robert J Marzano. *Transforming Classroom Grading.* ERIC, 2000.

[13] Kathleen Montgomery. Classroom rubrics: Systematizing what teachers do naturally. *The clearing house*, 73(6):324–328, 2000.

[14] Jack Snowman and Rick McCown. *Psychology Applied to Teaching.* Nelson Education, 2011.

MUIPC and Intent to Change IoT Privacy Settings*

C. Bryan Foltz[1] *and Laura Foltz*[2]
[1]*Management, Marketing, and Information Systems*
College of Business and Global Affairs
University of Tennessee at Martin
Martin, TN 38238
cfoltz1@utm.edu
[2]*Finance and Administration*
University of Tennessee at Martin
Martin, TN 38238
lfoltz@utm.edu

Abstract

The Internet of Things (IoT) has grown rapidly in recent years; along with this rapid growth have come multiple privacy concerns. This research utilizes the Mobile Users' Information Privacy Concerns (MUIPC) instrument [27] to examine user intention to change privacy settings on IoT devices.

1 Introduction

Privacy is one of the key ethical issues facing the information age [15]. Ongoing changes in technology have only increased the number of privacy threats [2]; as a result, privacy models must evolve to address these new threats and technologies.

The proliferation of the Internet of Things (IoT) presents a huge threat to user privacy, especially given the rapid increase in usage. These devices often incorporate a variety of sensors [28] and collect an amazing amount of

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

data including audio, video, location, and medical data; this data collection poses a threat to privacy [16]. Such data has been utilized in murder trails and, as Morrow [16] notes, Internet-connected cameras could also be subject to attacks.

Most IoT devices do allow users to alter privacy settings; unfortunately, many users have differing levels of privacy concerns. Some may seek to protect their privacy by altering settings, while others may not. As a result, privacy models must evolve to consider the impact of technological changes, user privacy concerns, and user decisions regarding changing privacy settings.

Solutions to the IoT privacy issue must consider user attitudes and awareness [28]. The current research utilizes the Mobile Users' Information Privacy Concerns (MUIPC) instrument [27] to examine user concern for privacy and intention to change settings within the IoT environment. Foltz and Foltz [9] demonstrated that the MUIPC model is applicable in determining intention to use the IoT; changing IoT privacy settings appears to be a related behavior for which the MUIPC might also apply.

2 Literature Review

2.1 Privacy

Privacy has been examined in multiple disciplines for quite some time [15] [2] [5]; interest in this concept has only increased with the rapid growth of technology [5] [2]. Privacy has been defined in multiple ways, with variations based on factors such as culture and legal concerns [14]. Many privacy definitions incorporate the idea of control over the use and secondary use of personal information [2]. Westin [24] provided an early definition of privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others." Later Westin [25] modified this definition to "the right to select what personal information about me is known to what people." More recent definitions also incorporate this concept of control. Belanger and Crossler [2] define privacy as "the desire of individuals to control or have some influence over data about themselves" while Sfar et al [19] add the collection, use, and sharing of data.

Within the extant information systems literature, multiple behavioral models utilize information privacy concern as a proxy for privacy [26]. Although multiple definitions have been offered, the current research will utilize the Xu et al [27] definition of information privacy concerns as "concerns about possible loss of privacy as a result of information disclosure to a specific external agent."

2.2 The Internet of Things (IoT

The Internet of Things (IoT) consists of a collection of interconnected devices that are also connected to the Internet [28]. These devices integrate the physical and digital worlds while collecting data without human interaction [23]. These IoT devices have gained rapid acceptance; Weinberg et al [23] predicted 50 billion such devices in use by 2020.

2.3 Privacy and the Internet of Things

Despite the rapid growth of the IoT, multiple privacy concerns remain. These devices collect a great deal of data [23] but how will that data be stored and used? Who owns the data [23] [28]?

Users surround themselves with these sensor-equipped devices which "aggressively collect personally identifiable information" about daily activities to help predict user needs [13]. Further, this information is shared between devices and services over the Internet [19] and creates multiple security and privacy concerns [19] [28]. Many users are simply unaware of this data collection and sharing and fail to read privacy policies [1]; each device presents another opportunity for privacy violations [17].

2.4 Changing Settings

Most IoT devices and related services offer options to alter a variety of settings. This allows the user some control over the collection and sharing of data. To help understand user privacy concerns and intention to change the associated settings, an existing survey (MUIPC) will be modified and examined within the IoT setting.

3 Methodology

3.1 Information Privacy Concern Measurement

Multiple privacy instruments exist within the extant information systems literature. Preibusch [18] provides an excellent summary of these. In 2012, Xu et al proposed the Mobile Users Information Privacy Concern (MUIPC) model which is based upon Communications Privacy Management (CPM) theory [27] and has been extended to the IoT [9]. Unfortunately, none of these instruments target user alteration of privacy settings; however, a recent study [8] applied the Dinev and Hu [6] awareness model of human behavior to this area: that model did not hold.

This research seeks to apply the MUIPC (Figure 1) to user alteration of privacy settings within the IoT environment. The MUIPC model consists of

three dimensions: perceived intrusion, perceived surveillance, and secondary use of information.

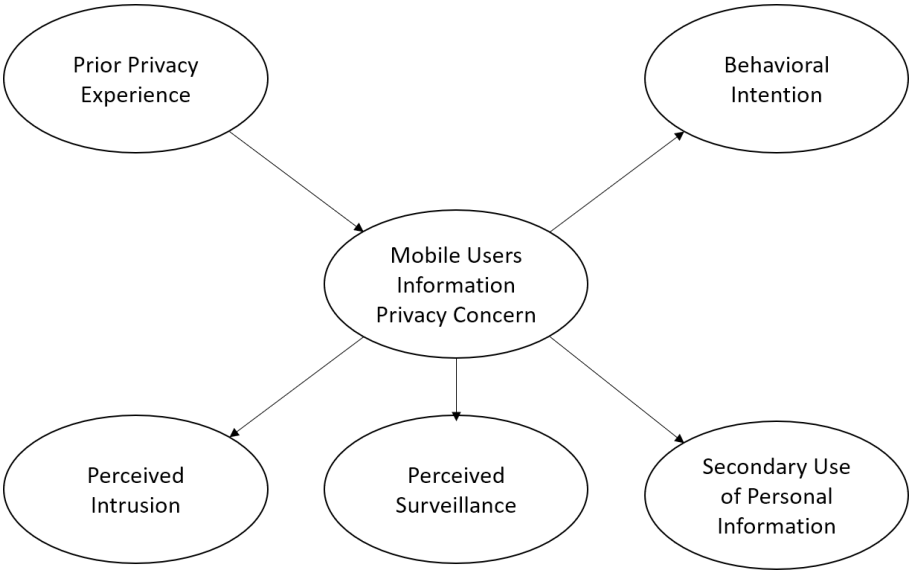


Figure 1: MUIPC Instrument [27]

3.2 Perceived Intrusion

Perceived intrusion reflects the ability of the data recipient to make decisions regarding the user's data [27]. Since IoT devices are designed to gather data using various sensors and to report that data over a network, the idea of intrusion exists for IoT devices. The CFIP captures this concept with the improper access and error dimensions while the IUIPC captures intrusion with the control dimension [27]. MUIPC captures user concern regarding physical and informational space [27].

3.3 Secondary Use of Information

Users are often concerned with the possibility of unauthorized use of their data [22] or the use of that data for a different purpose [21]. This concern is addressed within the CFIP, IUIPC, and MUIPC and is the same within the IoT setting.

3.4 Information Privacy Concern

Information Privacy Concern is a second-order factor. Xu et al [27] selected a second-order factor model as providing the most parsimonious fit.

3.5 Prior Privacy Experience and Behavioral Intention

The MUIPC model incorporated Prior Privacy Experience and Behavioral Intention to assist in validating the higher-order factor structure of the model. Prior Privacy Experience utilizes items from Smith et al [21] to measure respondents' past experiences with privacy violations. Behavioral Intention reflects the likelihood of an individual performing a specific behavior; individuals with prior privacy violations and higher privacy concerns are less likely to share information and should therefore be more interested in changing privacy settings. The current research utilizes survey items derived from [6] and modified by [8] to capture intention to change settings. These factors are all defined in Table 1.

3.6 Hypotheses

This research seeks to extend the MUIPC instrument within the Internet of Things to examine user intent to change IoT settings.

H1: Perceived Surveillance is correlated with Information Privacy Concern.

H2: Perceived Intrusion is correlated with Information Privacy Concern.

H3: Secondary Use of Personal Information is correlated with Information Privacy Concern.

H4: Prior Privacy Experience Influences Information Privacy Concern

H5: Information Privacy Concern influences Behavioral Intention

3.7 Methodology

Students at a mid-south Carnegie Master's level university were invited to complete an online survey by their normal classroom instructors. An opportunity to enter a random drawing for three gift cards was also offered to respondents (this process was administered by an independent department chair). 192 students, or 15.3 percent, of the 1257 students invited elected to complete the survey. Table 2 present descriptive statistics. The survey designed to gather this data contained the items from Dinev and Hu [6] and the MUIPC [27] instruments along with items measuring intention to change privacy settings and intention to use IoT devices [8] [9]. The Dinev and Hu model examining user alteration of privacy settings was only partially supported [8]. The MUIPC model examining the intention to utilize IoT devices was supported [9]. Can MUIPC also be used with the intention to change privacy settings? This study

Table 1: Definition of Key Constructs

Variable	Definitions
Perceived Surveillance (PS)	Composite of three items reflecting respondent perception of IoT devices "watching, listening to, or recording" [22]. Scoring ranged from 1 (strongly agree) to 7 (strongly disagree).
Perceived Intrusion (PI)	Composite of three items reflecting respondent perception of IoT devices creating a "violation of personal space, presence, or activities" [22]. Scoring ranged from 1 (strongly agree) to 7 (strongly disagree).
Secondary Use of Information (SUI)	Composite of three items reflecting respondent concern with the use of data collected by IoT devices for other purposes [21]. Scoring ranged from 1 (strongly agree) to 7 (strongly disagree).
Prior Privacy Experience (PPE)	Composite of three items reflecting respondents' past experience with privacy violations [27]. Scoring ranged from 1 (no instances) to 7 (over 10 instances).
Behavioral Intention (BI)	Composite of three items reflecting respondents' intention to change privacy settings on IoT devices [6]. Scoring ranged from 1 (strongly agree) to 7 (strongly disagree).
Information Privacy Concern (IPC)	Second—order factor composed of Perceived Surveillance, Perceived Intrusion, and Secondary Use of Information.

Table 2: Descriptive Statistics

Gender		Age		Internet and IoT Experience		
Gender	Freq.	Age	Freq.	Time Frame	Internet Freq.	IoT Freq.
Female	112	18-24	150	Less than a year	3	27
Male	68	25-34	14	1 – less than 2 years	1	26
Prefer not to say	5	35-44	11	2 – less than 3 years	2	21
		45-54	7	3 – less than 4 years	9	29
		55-64	3	4 – less than 5 years	7	13
		Over 65	0	5 – less than 6 years	12	11
				6 – less than 7 years	15	6
				More than 7 years	136	53

seeks to answer that question by utilizing factors and data from the existing survey.

SmartPLS was used to validate the measurement model and to test hypotheses. This software uses a variance-based approach for parameter estimation and a least-squares process to reduce demands on the measurement scales and sample size [4] [7] [10].

The measurement model was examined by comparing discriminant validity, convergent validity, and internal consistency reliability (ICR) to established heuristics. The model did meet the threshold for discriminant validity: the AVE values all exceeded the recommended .707 threshold and the square roots of the AVE exceeded the correlations among construct pairs [11]. The individual items loaded more heavily upon the expected constructs than on other constructs [12]. Tables 3 and 4 present the AVE values, correlations, and loadings; discriminant validity is supported.

Convergent validity is also supported, as indicated in Tables 3 and 4. The factor loadings all load significantly upon their respective constructs and exceed the recommended cutoff value of .70 [3].

Further, The Internal Consistency Reliability (ICR) values, which are analogous to Chronbach's Alpha, range from .88 to .93 and all exceed the accepted

Table 3: Internal Consistency Reliability, Convergent Validity, and Discriminant Validity

	ICR	AVE	PS	PI	SUI	PPE	BI	IPC
PS	.91	.77	**.88**					
PI	.92	.80	.86	**.89**				
SUI	.92	.81	.86	.83	**.90**			
PPE	.88	.71	.76	.76	.75	**.84**		
BI	.93	.81	.70	.70	.70	.59	**.90**	
IPC	.92	.80	.92	.95	.81	.75	.67	**.89**

Internal Consistency Reliabilities (in ICR column),
Average Variance Extracted (in AVE column),
AVE square roots (on diagonal),
and correlations (below diagonal).

heuristic value of .70 as suggested by [11]; further, the AVE values range from .71 to .81 which exceeds the heuristic of .5 [11]; the analysis thus supports internal consistency reliability.

The first three hypotheses are all significant at the .001 level. These hypotheses focus upon the relationships between the individual factors (perceived intrusion, perceived surveillance, and secondary use of personal information) and the second-order factor of information privacy concern. Thus, the MUIPC model does work within the IoT environment.

The remaining hypotheses (H4 and H5) are also supported at the .001 level. H4 examines the relationship between PPE and IPC, while H5 examines the relationship between IPC and BI. Prior privacy violation experiences influence privacy concern, and privacy concern is related to the intention to change privacy settings for IoT technology.

The IoT is growing rapidly; many consumers are unaware of the various privacy concerns created by this technology. A growing body of research suggests that this concern is indeed warranted. The risks posed by IoT technology can be mitigated by altering IoT privacy settings. The current research suggests that IoT users are becoming aware of these concerns and are beginning to change their settings to address these concerns.

4 Results

The path coefficients and t-statistics are presented in Table 5. All hypotheses are significant at an alpha level of .001 which suggests the MUIPC is useful in evaluating user intention to change IoT security settings.

Table 4: Factor Loadings

	PS	PI	SUI	PPE	IPC	BI
PS1	**.871**	.6743	.6813	.6196	.8345	.5411
PS2	**.899**	.7966	.8019	.6949	.8225	.5743
PS3	**.8552**	.8015	.7922	.6753	.7596	.7366
PI1	.8032	**.9231**	.7458	.6858	.914	.6254
PI2	.8061	**.9379**	.7561	.6945	.9306	.626
PI3	.6963	**.8106**	.7459	.6599	.6681	.6392
SUI1	.7643	.7154	**.9213**	.6348	.7113	.6099
SUI2	.7665	.7686	**.8585**	.6659	.7195	.6868
SUI3	.8081	.7723	**.9257**	.718	.7719	.5992
PPE1	.636	.6491	.6863	**.8744**	.645	.4433
PPE2	.6744	.6372	.5839	**.8008**	.6626	.5714
PPE3	.5871	.6218	.6086	**.8427**	.5618	.4625
BI1	.6092	.5955	.5944	.5122	.5584	**.9248**
BI2	.6593	.6957	.669	.5605	.6758	**.8607**
BI3	.6082	.5747	.5976	.5039	.5483	**.9088**

5 Limitations

There are numerous limitations within this research. First, a convenience sample of university students was utilized; this sample may not represent the general population. Further, most respondents reported significant experience with the Internet and IoT. A larger, more diverse sample could produce different results. Finally, this research combined elements and data from two earlier studies to evaluate the data in a new way.

Table 5: Path Coefficients and t-Statistics

	Path Coeff.	t-Statistics
H1: PS -> IPC	.92	26.2680*
H2: PI -> IPC	.95	125.8856*
H3: SUI -> IPC	.81	12.0578*
H4: PPE -> IPC	.75	10.7830*
H5: IPC -> BI	.67	7.0544*

* Signifacant at alpha = .001

6 Conclusions and Implications

The rather amazing growth of the IoT has resulted in users placing Internet-connected devices equipped with multiple sensors throughout their homes and environments; these devices collect an amazing amount of information and pose a threat to privacy [16]. While users often express a concern with privacy, these actions contradict those statements—this is termed the privacy paradox [20]. This paradox suggests a need to further understand privacy concerns within the IoT environment, especially a need to understand user intention to alter IoT privacy settings.

Although a plethora of privacy research exists, little effort has been expended examining the applicability of existing instruments to the IoT, especially regarding users changing the IoT privacy settings. This research extends the MUIPC to examine user alteration of privacy settings within the IoT environment, thus opening another avenue for future research.

Also, this research establishes the influence of prior privacy concerns upon user alteration of IoT settings. Understanding why users do or do not alter privacy settings is an important factor in understanding privacy within the IoT environment.

Future research should examine user awareness of privacy risks within the IoT environment; the privacy paradox suggests that user behavior and attitudes may not match. This could be explained by a lack of user understanding regarding the data collected and shared by IoT devices, or by user failure to alter available privacy settings. Additional research should examine the impact of training and education upon this privacy paradox: perhaps greater knowledge and experience would allow users to more accurately understand and protect their data. Finally, a better understanding of user privacy concerns and willingness to alter settings should allow IoT manufacturers to offer instructions or options to help mitigate this loss of privacy.

References

[1] Christine Bannan. The iot threat to privacy. *TechCrunch, available at: https://techcrunch.com/2016/08/14/the-iot-threat-to-privacy/(accessed 2019)*, 2016.

[2] France Bélanger and Robert E Crossler. Privacy in the digital age: a review of information privacy research in information systems. *MIS quarterly*, pages 1017–1041, 2011.

[3] Edward G Carmines and Richard A Zeller. *Reliability and validity assessment*, volume 17. Sage publications, 1979.

[4] Wynne W Chin et al. The partial least squares approach to structural equation modeling. *Modern methods for business research*, 295(2):295–336, 1998.

[5] Tamara Dinev and Paul Hart. Internet privacy concerns and their antecedents-measurement validity and a regression model. *Behaviour & Information Technology*, 23(6):413–422, 2004.

[6] Tamara Dinev and Qing Hu. The centrality of awareness in the formation of user behavioral intention toward protective information technologies. *Journal of the Association for Information Systems*, 8(7):23, 2007.

[7] R Frank Falk and Nancy B Miller. *A primer for soft modeling.* University of Akron Press, 1992.

[8] C. Bryan Foltz and Laura Foltz. Privacy, awareness, and the internet of things. In *Proceedings of the Decision Sciences Institute Annual Conference*. Decision Sciences Institute, 2019.

[9] C Bryan Foltz and Laura Foltz. Mobile users' information privacy concerns instrument and iot. *Information & Computer Security*, 2020.

[10] Claes Fornell and Fred L Bookstein. Two structural equation models: Lisrel and pls applied to consumer exit-voice theory. *Journal of Marketing research*, 19(4):440–452, 1982.

[11] Claes Fornell and David F Larcker. Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research*, 18(1):39–50, 1981.

[12] David Gefen and Detmar Straub. A practical guide to factorial validity using pls-graph: Tutorial and annotated example. *Communications of the Association for Information systems*, 16(1):5, 2005.

[13] Hosub Lee and Alfred Kobsa. Understanding user privacy in internet of things environments. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 407–412. IEEE, 2016.

[14] Naresh K Malhotra, Sung S Kim, and James Agarwal. Internet users' information privacy concerns (iuipc): The construct, the scale, and a causal model. *Information systems research*, 15(4):336–355, 2004.

[15] Richard O Mason. Four ethical issues of the information age. *MIS quarterly*, pages 5–12, 1986.

[16] S. Morrow. 5 reasons privacy and iot are incompatible. `https://www.iotforall.com/five-reasons-privacy-iot-incompatible/`.

[17] Swaroop Poudel. Internet of things: underlying technologies, interoperability, and threats to privacy and security. *Berkeley Technology Law Journal*, 31(2):997–1022, 2016.

[18] Sören Preibusch. Guide to measuring privacy concern: Review of survey and observational instruments. *International Journal of Human-Computer Studies*, 71(12):1133–1143, 2013.

[19] Arbia Riahi Sfar, Enrico Natalizio, Yacine Challal, and Zied Chtourou. A roadmap for security challenges in the internet of things. *Digital Communications and Networks*, 4(2):118–137, 2018.

[20] H Jeff Smith, Tamara Dinev, and Heng Xu. Information privacy research: an interdisciplinary review. *MIS quarterly*, pages 989–1015, 2011.

[21] H Jeff Smith, Sandra J Milberg, and Sandra J Burke. Information privacy: measuring individuals' concerns about organizational practices. *MIS quarterly*, pages 167–196, 1996.

[22] Daniel J Solove. A taxonomy of privacy. *U. Pa. L. Rev.*, 154:477, 2005.

[23] Bruce D Weinberg, George R Milne, Yana G Andonova, and Fatima M Hajjat. Internet of things: Convenience vs. privacy and secrecy. *Business Horizons*, 58(6):615–624, 2015.

[24] A. F. Westin. *Privacy and Freedom*. Athenaum, New York, 1967.

[25] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[26] Heng Xu, Tamara Dinev, H Jeff Smith, and Paul Hart. Examining the formation of individual's privacy concerns: Toward an integrative view. *International Conference on Information Systems 2008 Proceedings*, page 6, 2008.

[27] Heng Xu, Sumeet Gupta, Mary Beth Rosson, and John M Carroll. Measuring mobile users' concerns for information privacy. *International Conference on Information Systems 2012 Proceedings*, 2012.

[28] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. User perceptions of smart home iot privacy. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–20, 2018.

Narcoleptic Philosophers: Message Delays on Paths with ON/OFF Nodes*

Timothy Nix[1], *Riccardo Bettati*[2]
[1]*Department of Computer Science*
Stephen F. Austin State University
Nacogdoches, TX 75962
timothy.nix@sfasu.edu
[2]*Department of Computer Science & Engineering*
Texas A&M University
College Station, TX 77843
bettati@tamu.edu

Abstract

We develop a model for message delivery between a source node and a destination node in which each node operates as a binary renewal processes; that is, nodes are only available for communication during a portion of the time. We determine the closed-form formula for calculating the expected message delay and the probability distribution for message delivery as a function of time. From there, we examine a message's delay from its source to its destination as it propagates from node to node along a path topology where each node can only communicate with its nearest two neighbors (one neighbor for each node at the ends of the path) and develop the probability distribution for the message delay.

1 Introduction

Two philosophers are sitting side-by-side. Unfortunately, both suffer from narcolepsy; that chronic sleep disorder that causes the afflicted to be overcome

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

by drowsiness and suddenly fall asleep. Thus, each philosopher alternates between random periods of being awake and random periods of being asleep. While awake (and, perhaps, even while asleep), each philosopher ponders the deep questions of the Universe. When one philosopher has a sudden brilliant insight, if he is awake, he immediately wishes to share his new found knowledge with his colleague. However, he can only do so if both he and his colleague are awake. How long until the first philosopher can successfully communicate his new idea to his colleague?

Understanding this form of message delay is important in many types of information networks, such as delay tolerant networks (DTN), mobile ad hoc networks (MANET), sensor networks, and distributed Internet of Things (IoT) networks. Message propagation and delay in these types of networks depend on two nodes that are in range of each other to be active (as opposed to dormant or in power-savings mode). Much of the research into these types of networks that are concerned with message delay also examine routing issues [1, 5], quality of service [6], and security [2, 3, 7].

2 A Description of the Basic Model

To model our narcoleptic philosophers, consider a communication network in which each philosopher is represented as a node within the network. The basic model consists of only two nodes, and a message, m, to be transmitted between the two nodes. The philosopher/node sending the message is the *source* node, and the *destination* node receives the message.

Each node is modeled as a *binary renewal process*; that is, it can be either ON or OFF, corresponding to the philosopher being awake or asleep, respectively. Each node is ON for an independent and identically distributed (i.i.d.) random duration drawn from an exponential distribution with parameter λ. Similarly, each node is OFF for an i.i.d. random duration drawn from an exponential distribution with parameter μ. A node will indefinitely cycle through repeated states of ON and OFF during its *life cycle* (as per Fig. 1). Let $\Delta t_{ON}(i)$ and $\Delta t_{OFF}(i)$ denote the duration that a node is ON and OFF respectively during the i^{th} iteration of its life cycle.

The message can only be *generated* while the source is ON. The destination can only receive the message while both source and destination are ON. The delay between message generation and delivery is the *message delay*, Δt_m. We assume that the transmission time (mostly propagation time) between the source and destination is zero. Thus, the message delay is strictly the delay until both source and destination are in the ON state.

The probability that a node is ON at some random time during its life cycle is $Pr(ON) = \mu/(\lambda + \mu)$ and the probability that the same node is OFF

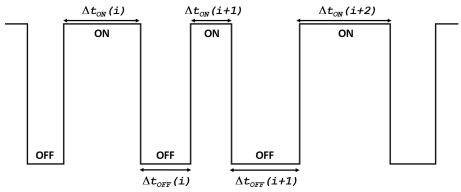


Figure 1: The Life Cycle of a Node

is $Pr(OFF) = \lambda/(\lambda + \mu)$.

We are interested in the amount of time it takes the *source* to communicate its message to the *destination*; that is, the cumulative probability distribution (CDF) of the message delay as a function of λ, μ, and time t.

3 Modeling Message Delay Using Stochastic Processes

We model the evolution of the generation and delivery of a message in a two-node system using a Markov chain, as shown in Fig. 2a. At message generation, we know that the source node is ON. The probability that the destination node is ON is independent of the source node. If both the source and destination nodes are ON, then the message is delivered immediately, and $\Delta t_m = 0$.

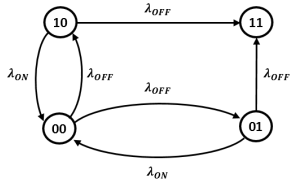
The more interesting case is when the destination node is OFF. We represent the process of transmitting the message from source to destination node using four states: State 10 represents when the source node is ON but the destination node is OFF. State 00 represents when both nodes are OFF, and State 01 represent when only the destination node is ON. State 11 represents when both source and destination are ON. Message delivery occurs only while the system is in State 11.

Each state transition is controlled by an exponential random variable, which is defined by its associated *transition rate*. For example, the transition rate from State 10 to State 00 corresponds to the rate of the source node transitioning from ON to OFF. This occurs based on the duration of the ON state, which is drawn from the exponential distribution with parameter λ. The same holds for the other transitions.

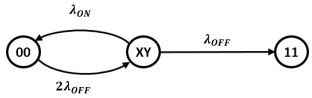
The probabilities associated with each transition is the transition rate divided by the sum of the transition rates of the originating state. Thus, the transition from either State 10 or State 01 to State 11 occurs with probability $Pr(ON)$, and the probability that the transition from either State 10 or State 01 to State 00 occurs is $Pr(OFF)$.

Notice that the Markov chain of Fig. 2a exhibits a symmetry: The transitions from State 00 to State 11 are identical whether going through State 10

or State 01. We can therefore simplify this Markov model by collapsing State 10 with State 01. We denote this combined state as State XY, where X and $Y \in 0, 1$ but $X \neq Y$. Our simplified Markov chain is shown in Fig. 2b.



(a) Full Markov Chain



(b) Simplified Markov Chain

Figure 2: Markov Chain Representation of one-hope Message Delivery

One way of viewing our simplified Markov chain is to recognize how the transition rates specified in Fig. 2b reflect the rate at which a node will change states. Thus, from State XY, whichever node is ON will remain ON for the duration specified by λ. Likewise, the transition from State XY to State 11 occurs when the node that is OFF switches to ON, which occurs based on the parameter μ. The transition from State 00 to State XY occurs when either of the two nodes switches to ON and occurs at a rate based on the sum of the two rate parameters, $\mu + \mu = 2\mu$.

3.1 Computing the Delay Distribution on a Single Hop

We compute the delay for *a single-hop communication given that the second node is OFF* at time t_0, based on the simplified Markov Chain in Fig. 2b. This simple system can be described as follows, with the Kolmogorov forward equations on the left and their Laplace transforms on the right:

$$\frac{dP_{00}}{dt} = -2\mu P_{00}(t) + \lambda P_{XY}(t) \qquad s\bar{P}_{00}(s) \qquad = -2\mu\bar{P}_{00}(s) + \lambda\bar{P}_{XY}(s)$$

$$\frac{dP_{XY}}{dt} = 2\mu P_{00}(t) - (\mu + \lambda)P_{XY}(t) \qquad s\bar{P}_{XY}(s) - 1 \quad = 2\mu\bar{P}_{00}(s) - (\mu + \lambda)\bar{P}_{XY}(s)$$

$$\frac{dP_{11}}{dt} = \mu P_{XY}(t) \qquad s\bar{P}_{11}(s) \qquad = \mu\bar{P}_{XY}(s) \quad .$$

If $f'(t)$ is the derivative of $f(t)$, then recall that the Laplace transform $\mathcal{L}$ of the $f'(t)$ is given by: $\mathcal{L}\{f'(t)\} = s\mathcal{L}\{f(t)\} - f(0)$. Since the starting state of the system is State *10* = State XY, we set $P_{XY}(0) = 1$, and $P_{00}(0) = 0$ and $P_{11}(0) = 0$. As a result, we get the above Laplace transform of the system.

Solving for $\bar{P}_{11}(s)$[1], we get

$$\bar{P}_{11}(s) = \frac{1}{s} \times \frac{\mu s + 2\mu^2}{(s+\alpha_1)(s+\alpha_2)} \quad . \tag{1}$$

where $\alpha_1, \alpha_2 = (3\mu + \lambda) \pm \sqrt{\mu^2 + 6\mu\lambda + \lambda^2}$.

We rewrite Eq. 1 and compute the inverse Laplace transform to get the CDF $F_1(t)$ for the one-hop delay *considering only the case when the destination node is OFF*:

$$F_1(t) = P_{11}(t) = \frac{2\mu^2}{\alpha_1\alpha_2} + \frac{(2\mu^2 - \alpha_1\mu)e^{-\alpha_1 t}}{\alpha_1(\alpha_1 - \alpha_2)} + \frac{(\alpha_2\mu - 2\mu^2)e^{-\alpha_2 t}}{\alpha_2(\alpha_1 - \alpha_2)} \quad . \tag{2}$$

The probability density function (pdf) for the message delay $f_1(t)$ under the same conditions can be easily derived by first representing its Laplace transform $\bar{f}_1(s) = s\bar{F}_1(s) - F_1(0)$ and then inverting the transform $\bar{f}_1(s)$ to get the density function:

$$f_1(t) = \frac{(\alpha_1\mu - 2\mu^2)e^{-\alpha_1 t} + (2\mu^2 - \alpha_2\mu)e^{-\alpha_2 t}}{\alpha_1 - \alpha_2} \quad . \tag{3}$$

Thus, we denote $D_1(t)$ to be the probability that the message has been delivered by time t under the condition that the destination node may or may not be OFF; that is, we represent the CDF of the 1-hop case as:

$$D_1(t) = Pr(\Delta t_m < t) = Pr(ON) + Pr(OFF) \times F_1(t) \quad , \tag{4}$$

where the message is delivered instantaneously with probability $Pr(ON)$ and in accordance with the special case CDF denoted by $F_1(t)$ with probability $Pr(OFF)$. In Fig. 4a, we see the plots of the CDFs derived from Eq. 4. We can use the CDF to determine the *expected message delay* over a single hop as $E[\Delta t_m] = \int_0^\infty 1 - D_1(t)dt = \frac{\lambda^2 + 2\lambda\mu}{2\mu^2(\lambda+\mu)}$.

4 Store and Forward Messaging on Multi-Hop Paths

Now, consider multiple narcoleptic philosophers sitting in the same row of an auditorium. The philosopher sitting on one end of the row suddenly has a

[1]We loosely follow the derivation of time-to-failure in the "two-component parallel-redundant system with a single repair facility of rate μ" discussed in [4].

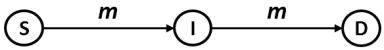


Figure 3: Three Nodes in a Path Topology

brilliant insight and wants to communicate his idea to his colleagues. The philosopher quietly shares his insight with his immediate neighbor as soon as both are awake. On receiving the information, each neighbor, in turn, quietly communicates the idea with to next neighbor as soon as both are awake. How long will it take to communicate this idea to the philosopher sitting at the opposite end of the row? In other words, what is the end-to-end delay of the message?

The message propagates through the path topology from the source node to the destination node one hop at a time whenever two adjacent nodes are both ON and exactly one of the two nodes "has" a copy of the message. The message is delivered when the destination node is ON and its sole neighbor is both ON and has a copy of the message.

4.1 The 2-Hop Case

We extend our model to a path with two hops as represented in Fig. 3 where Node S is the source node, Node D is the destination node, and Node I is an intermediate node. While ON, Node S generates the message and when both Node S and Node I are ON, then the message propagates to Node I. Once Node I has the message and is simultaneously ON with Node D, the message can then be successfully delivered.

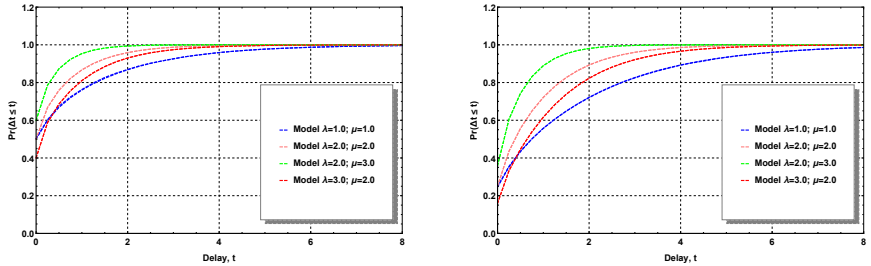
To represent the distribution of the end-to-end message delay within this system, we need to account for three cases: (1) all nodes are ON and the message is transmitted instantaneously; (2) exactly one of the two hops happens instantaneously; or (3) there is delay at both hops.

For Case (1), the delay is zero. For Case (2), the time delay is represented by Eqn. 3. For Case (3), the delay is the sum of the two independent random variables drawn from the distribution of the one-hop case. The density function is, thus, the convolution of Eq. 3 with itself; that is, $f_2(t) = f_1(t) * f_1(t)$. The CDF is calculated by integrating $f_2(t)$ to give us $F_2(t)$.

The CDF for the full system is the sum of the probability of each state occurring times the delay that each particular state would incur: Thus, the CDF $D_2(t)$ can be directly derived to be:

$$D_2(t) = Pr(ON)^2 + 2Pr(ON)Pr(OFF) \times F_1(t) + Pr(OFF)^2 \times F_2(t) \quad . \quad (5)$$

In Fig. 4b, we plot Eq. 5 under the same conditions as before but across two hops instead of one.



(a) One-hop Case (b) Two-hop Case

Figure 4: $Pr(\Delta t_m < t)$ on one and two hops, for varying λ and μ

4.2 Path Topologies with More than Three Nodes

Progressing from the one-hop case to the two-hop case, we see the emergence of the pattern for the n-hop case. We need to define the density function for each case of the message being delayed at 0, 1, 2, ... n nodes. As such, we define the following recurrence relation:

$$f_k(t) = \begin{cases} 1 & \text{for } k = 0 \\ f_1(t) & \text{for } k = 1 \\ f_{k-1}(t) * f_1(t) & \text{for } k \geq 2 \end{cases}$$

where k is the number of hops that incur a delay. So when $k = 0$, the message travels from source node to destination node instantaneously. If delay occurs on a single hop, then Eq. 3 is the density function of the delay. For $k \geq 2$, the repeated convolution with $f_1(t)$ produces the density function for non-zero delay occuring over exactly k hops (with no delay over $n - k$ hops). Likewise, $F_k(t) = \int_0^\infty f_k(t)dt$ to give us the cumulative density over k hops.

The resulting CDF $D_n(t)$ for the n-hop delay then naturally follows to be:

$$D_n(t) = \sum_{k=0}^{n} \binom{n}{k} Pr(ON)^{n-k} Pr(OFF)^k F_k(t) \quad . \tag{6}$$

The value of $D_n(t)$ in Eq. 6 gives the probability that the message has traversed n hops at time t. The resulting *expected message end-to-end delay* is the sum of n single-hop expectations, or $E[\Delta t_m] = n \times \frac{\lambda^2+2\lambda\mu}{2\mu^2(\lambda+\mu)}$.

5 Conclusion and Future Work

In this paper, we presented the probability distribution for a message traveling from a source node to a destination node along intermediate nodes within a

sequential path in which the message can only be transmitted when: (1) one of two adjacent nodes possesses the message and the other does not; and (2) both nodes are available for communication.

In the sequel, we will extend the distribution model to network topologies more complex than the path topology examined in this paper. Also, we will look for real-world examples and compare the distributions generated by the real-world examples with our distribution model.

References

[1] F. Ciucu, O. Hohlfeld, and P. Hui. Non-asymptotic throughput and delay distributions in multi-hop wireless networks. In *2010 48th Annual Allerton Conf. on Communication, Control, and Computing (Allerton)*, pages 662–669, Sep. 2010.

[2] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *Proc. of the 1st ACM Int. Workshop on Quality of Service & Security in Wireless and Mobile Networks*, Q2SWinet '05, page 16–23, New York, NY, USA, 2005. ACM.

[3] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Information Assurance in Sensor Networks. In *Proc. of the 2nd ACM Int. Conf. on Wireless Sensor Networks and Applications*, WSNA '03, page 160–168, New York, NY, USA, 2003. ACM.

[4] D. P. Gaver. Failure Time for a Redundant Repairable System of Two Dissimilar Elements. *IEEE Trans. on Reliability*, R-13(1):14–22, 1964.

[5] Robin Groenevelt, Philippe Nain, and Ger Koole. Message Delay in MANET. *SIGMETRICS Perform. Eval. Rev.*, 33(1):412–413, June 2005.

[6] Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. Adaptive clock synchronization in sensor networks. In *Proc. of the 3rd Int. Symp. on Information Processing in Sensor Networks*, IPSN '04, page 340–348, New York, NY, USA, 2004. ACM.

[7] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Data Transmission in Mobile Ad Hoc Networks. In *Proc. of the 2nd ACM Workshop on Wireless Security*, WiSe '03, page 41–50, New York, NY, USA, 2003. ACM.

Towards a Software System for Spatio-Temporal Authorization

Mustafa Al Lail, Marshal Moncivais, and Miguelangel Trevino
School of Engineering
Texas A&M International University
Laredo, TX 78041
mustafa.allail@tamiu.edu
marshalmoncivais@dusty.tamiu.edu
miguelangeltrevino@dusty.tamiu.edu

Abstract

The increasing dependency on digital technology has made the concept of data security an important concern. Not only how information is accessed, but also where and when have become important considerations in cyber-security. Certain situations exist where it is necessary to restrict access based on time and location. An example is a policy for a medical institution where doctors can only access patient records at hospitals during their shifts. The Generalized Spatio-Temporal Role-Based Access Control model (GSTRBAC) determines users' access to resources based on such information. This paper describes a software architecture and its current implementation of the GSTRBAC model.

1 Introduction

Modern society has evolved quickly with the advent of the Internet and electronic resources. This has created a shift in terms of how companies, governments, and organizations conduct their business and access and use their resources. However, this new method of resource access has created new ways for attackers to gain access to confidential resources. According to a recent study, there had been at least 5,183 breaches, exposing 7.9 billion records in 2019 alone [5]. Such breaches pose a significant challenge to society.

A potential cause of the rise in breaches is the increased use of mobile applications with weak security measures. Nowadays, people use their mobile devices to gain access to governmental and business services. Traditionally, to access such services, the users' requests need to be authenticated and authorized.

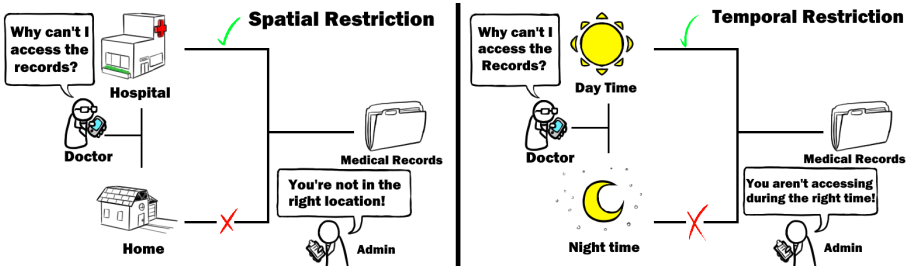


Figure 1: The effect of location and time on resource access.

The current authorization methods are commonly based on the use of the role-based access control (RBAC) model [8]. RBAC uses the users' credentials (based on their roles in an organization) to determine access to resources. However, there are applications (e.g., the telemedicine application iMediK [6]) that require more than standard users' credentials (i.e, the factors of time and location). Consider Figure 1 for an illustration of such a system. In such a system, access control policies state that doctors can only access patients' records when they are in the hospital and during their shift; otherwise, they should be denied.

The RBAC model lacks support for time and location constraints. Towards this end, researchers have developed new models to accommodate spatio-temporal requirements [2, 4, 7, 9]. The Generalized Spatio-Temporal Role-Based Access Control model (GSTRBAC) is an extension of RBAC [1] that integrates time and location into authorization decisions.

In this paper, we discuss the progress of an undergraduate research project that aims to provide an implementation of a software system that incorporates Spatio-temporal data when determining access to resources. The implementation is based on the abstract GSTRBAC model [1]. We use the Unified Modeling Language (UML) [3] (the standard modeling language in the software industry) to define a software architecture of a system implementing GSTRBAC policies. Additionally, we discuss our implementation of the architecture using the C# programming language. The result of this project is a system that demonstrates the feasibility of GSTRBAC policies.

This paper provides an overview of the GSTRBAC model in Section 2. In Section 3, we discuss the software architecture and its implementation of the GSTRBAC system. Section 4 concludes the paper and points out future work.

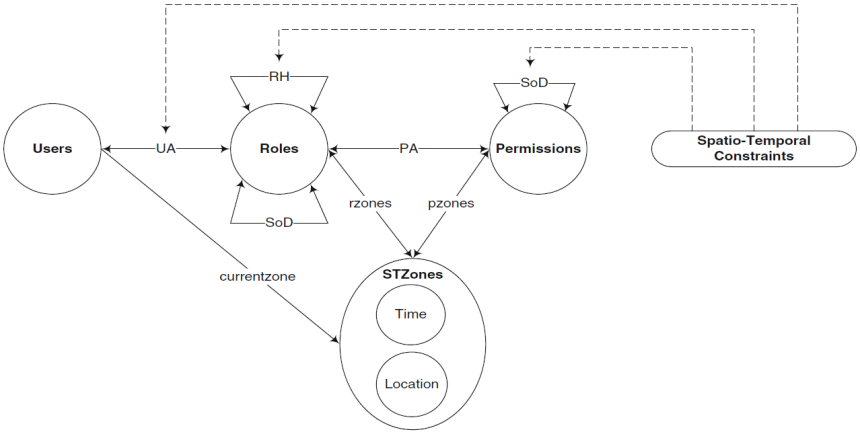


Figure 2: The conceptual model of the generalized spatio-temporal RBAC.

2 Overview of GSTRBAC

Figure 2 shows the conceptual model of GSTRBAC [1]. You can see five main components of the model: Users, Roles, Permissions, STZones, and Spatio-Temporal constraints. These components represent the set of users, roles, permissions, spatio-temporal zones, and constraints that exist in a system, respectively. Users are assigned to roles (denoted by UA) that specify job positions. Similarly, each role has different permissions assigned to it (denoted by PA) that determine the access granted to the role and, in turn, to the users. The separation of user and role assignments eases the management of access policies. The GSTRBAC model uses spatio-temporal zones (denoted by STZones) to define where and when a user is assigned to a role or where and when a permission is accessible. Location data comes in two forms: physical and logical. A physical location would be a specific point, like GPS coordinates, while a logical location would be an abstract location like Fort Worth, TX. Time data is represented by intervals such as 9:00 AM-5:00 PM.

As you can see in Figure 2, users, roles, and permissions have relations with the set of STZones (i.e., currentzone, rzones, and pzones). These relations indicate where and when these sets are enabled. For example, the pzones relation indicates where permissions can be activated. Similarly, roles can only be assigned to or activated by a user at specific spatio-temporal zones. The GSTRBAC model also supports advanced access control concepts such as role hierarchy (denoted by RH) and separation of duty (denoted by SoD) between roles and permissions. Interested readers of these concepts can find more information in the GSTRBAC model paper [1].

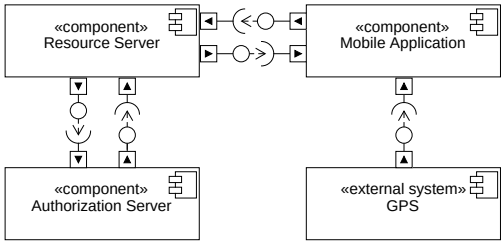


Figure 3: Software architecture of the GSTRBAC system.

3 Software Architecture and Implementation

We use the UML component diagram to model the software architecture of a system implementing the GSTRBAC policies [3]. The architecture is comprised of three main components, as seen in Figure 3, in addition to an outside external GPS component linked to the Mobile Application for location tracking purposes. These components communicate with each other to determine access to resources and to distribute them following the steps displayed in Figure 4.

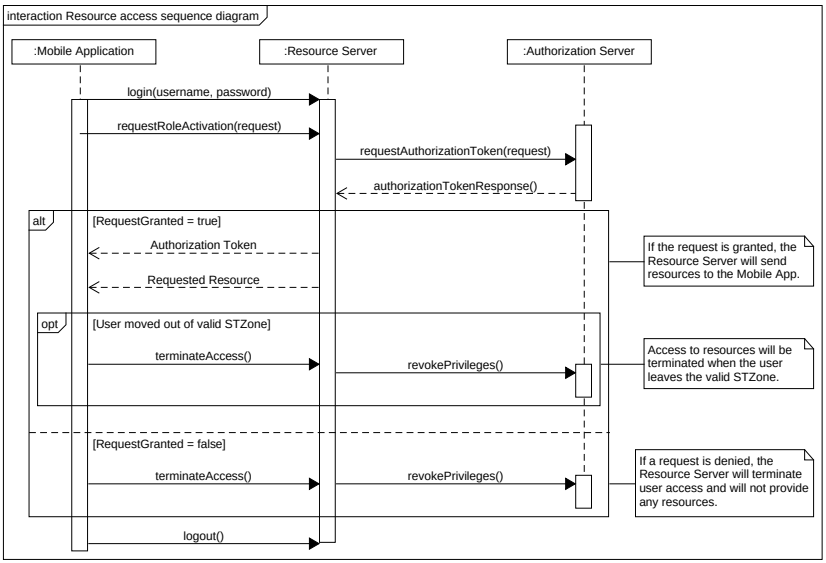


Figure 4: UML sequence diagram for resource access communication protocol.

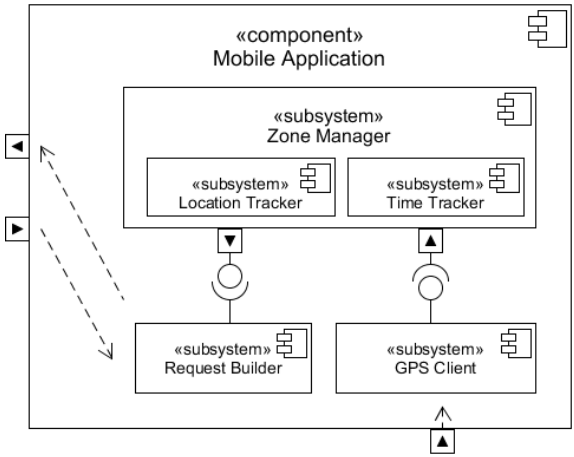
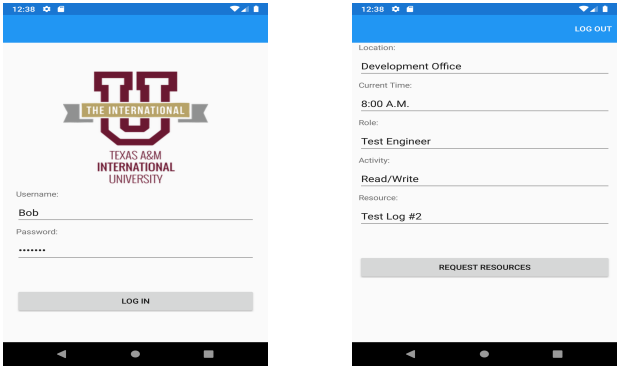


Figure 5: Mobile application software subcomponents.

3.1 Mobile Application

The Mobile Application component is the conduit through which the user logs into the system and requests and accesses resources. After a user logs in, this module compiles the user's spatio-temporal data along with the requested resources and sends them to the Resource Server. Figure 5 shows that the Mobile Application includes the Request Builder subcomponent to prepare an access request. The Request Builder compiles STZone data obtained from the Zone Manager. The Zone Manager collects the user's time and location data from the Time Tracker and the Location Tracker. The Location Tracker is linked to a GPS Client that is designed to receive GPS data from the user's device. The request package (created by the Request Builder) is then sent to the Resource Server component. If access is granted, the Zone Manager continually checks the STZone data to ensure the user has not moved out of the authorized zone. If the user is no longer in the authorized zone, the system terminates access and revokes the user's privileges to the required resources, as shown in Figure 4.

The Mobile Application currently consists of the main component class named MobileAppClient integrated with a graphical user interface (GUI) application implemented using a Xamarin Form framework to build a mobile application for Android and iOS. The MobileAppClient class includes the subcomponents shown in Figure 5. The subcomponents are further refined and implemented by smaller subcomponents (not shown in the figure). The Xamarin application GUI consists of many classes to demonstrate the functionality



(a) Login window. (b) Request window.

Figure 6: Android mobile application user-interface.

of the mobile client from the users' perspective. Figure 6 displays the current Xamarin Android handset application. Users enter their credentials to log in and select a role, an activity to perform, and a resource to request. The request is then packaged as a serialized object and sent to the Resource Server through a network socket. The current Mobile Application has a full implementation of the subcomponents, except for the GPS Client that has not been fully implemented.

Note that to demonstrate the feasibility of the approach without the need to physically move around different STZones, the current application allows the user to select a current location and time. These parameters should be fetched directly by the mobile application itself, not supplied by the user. The application is tamper-proof and it periodically should fetch the location and the time of the user using the current time of the system and the GPS client. This ensures that the correct location and time are being reported by the application and can not be spoofed.

3.2 Resource Server

The Resource Server is the intermediary server between the Mobile Application and Authorization Server components, as seen in Figure 3. Figure 7 shows the internal subcomponents of the Resource Server. When a user sends a login message, it includes the user's credentials in the request. The Credential Evaluator is an authentication service that validates users' credentials by comparing them with stored credentials in the Resource Database. If the credentials are validated, the system proceeds to the authorization phase; otherwise, the request

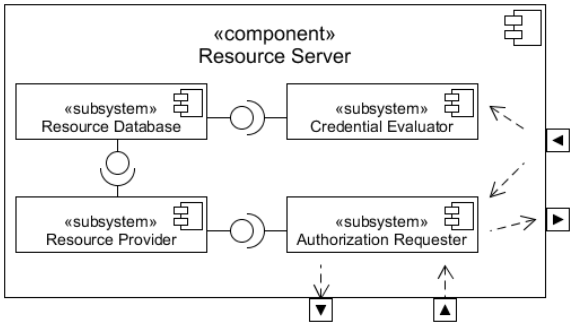


Figure 7: Resource server software subcomponents.

is denied.

The Authorization Requester requests an authorization token (from the Authorization Server) indicating if the Resource Server is allowed to provide the requested resources to the user. This token contains information regarding the user, its roles, allowed permissions, and the spatio-temporal zone. While this token is active, the Resource Provider subcomponent provides access to the requested resources by fetching it from the Resource Database. This token remains active until the user logs out, or moves to an invalid spatio-temporal zone. When a token expires, the Resource Server requests the updated spatio-temporal data from the Mobile Application component to request a new token from the Authorization Server component. If the token is granted again, the user continues to have access to the requested resources without interruption. If the token is denied, however, the resource access is immediately revoked from the user, as shown in Figure 4.

The current Resource Server has a full implementation of the four subcomponents classes. As shown in Figure 7, these classes are the Credential Evaluator, Resource Provider, Authorization Requester, and Resource Database. Credential Evaluator is responsible for the verification of user credentials like username and password, though it does not check the spatio-temporal zone of the user. The class Authorization Requester prepares and sends a request to the Authorization Server to request an authorization token. Finally, the Resource Provider class examines what activity and resource the user has selected and requests the appropriate resources from the Resource Database to send to the user, assuming that a token has been granted by the Authorization Server.

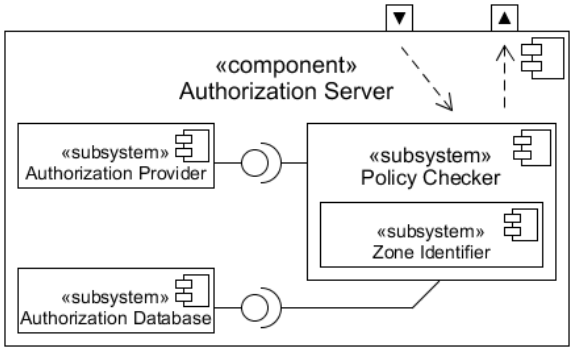


Figure 8: Authorization server software subcomponents.

3.3 Authorization Server

Figure 8 shows the Authorization Server subcomponents that are responsible for policy evaluation and token generation. The Policy Checker receives STZone data from the Resource Server and then interprets it by sending it to Zone Identifier, a subcomponent of Policy Checker. Zone Identifier checks the Authorization Database to see if the STZone of the user is authorized to access the requested resources. The Authorization Database, which stores access control policies for the model, is consulted to determine access. Based on the authorization result, the Authorization Provider subcomponent then distributes an authorization-token to the Resource Server. The Authorization Server subcomponents are fully implemented, in addition to a separate web application for administering the authorization database.

4 Conclusion

In this paper, we reported the results of an undergraduate research project in which we created a software architecture and its implementation of the GSTRBAC model. The GSTRBAC model is an authorization technique reliant on the factors of space and time. We created the various components of the architecture to ensure that resources remain secure while adhering to spatio-temporal constraints. The current implementation shows how spatio-temporal access constraints can be integrated with a software system to achieve a higher level of authorization. Future work will involve enhancing the quality of the software by adding features related to different quality attributes such as usability, reliability, and security.

5 Acknowledgement

This research was supported by the National Science Foundation grant HRD-1911375 and the Act-On-Idea program at Texas A&M International University.

References

[1] Ramadan Abdunabi, Mustafa Al-Lail, Indrakshi Ray, and Robert B France. Specification, validation, and enforcement of a generalized spatio-temporal role-based access control model. *IEEE Systems Journal*, 7(3):501–515, 2013.

[2] Nuray Baltaci Akhuseyinoglu and James Joshi. A risk-aware access control framework for cyber-physical systems. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 349–358. IEEE, 2017.

[3] Grady Booch, James E. Rumbaugh, and Ivar Jacobson. *The unified modeling language user guide - covers UML 2.0, Second Edition.* Addison Wesley object technology series. Addison-Wesley, 2005.

[4] Yan Cao, Zhiqiu Huang, Yaoshen Yu, Changbo Ke, and Zihao Wang. A topology and risk-aware access control framework for cyber-physical space. *Frontiers of Computer Science*, 14(4):1–16, 2020.

[5] Security Magazine. 7.9 billion records exposed so far in 2019. `https://www.securitymagazine.com/articles/91267-9-billion-records-exposed-so-far-in-2019`.

[6] Amiya K Maji, Arpita Mukhoty, Arun K Majumdar, Jayanta Mukhopadhyay, Shamik Sural, Soubhik Paul, and Bandana Majumdar. Security analysis and implementation of web-based telemedicine services with a four-tier architecture. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 46–54. IEEE, 2008.

[7] Konstantinos Rantos, Konstantinos Fysarakis, Charalampos Manifavas, and Ioannis G Askoxylakis. Policy-controlled authenticated access to lln-connected healthcare resources. *IEEE Systems Journal*, 12(1):92–102, 2018.

[8] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[9] Yingjie Xue, Jianan Hong, Wei Li, Kaiping Xue, and Peilin Hong. Labac: A location-aware attribute-based access control scheme for cloud storage. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

Use of Predefined Computing Blocks in Algorithmic Thinking*

S.R. Subramanya
Computer Science Department
University of Central Oklahoma
Edmond, OK 73034
ssubramanya@uco.edu

Abstract

The ultimate objectives of all the various facets of computing are about solving real–world problems to obtain the best (optimum, whenever possible) solutions, using minimum computing resources. Use of appropriate Data Structures and Algorithms is at the core of the above endeavor. Students and professionals in the Computing disciplines (Computer Science, Computer Engineering, Data Science, Information Systems, Artificial Intelligence, etc.) need to have a good grasp of algorithmic techniques and algorithmic thinking skills in order to effectively solve information–rich, information–driven, real–world and societal problems. Transforming a given problem to another known problem, for which a solution already exists, is an important skillset. Problem transformation skills are known to augment algorithmic thinking (based on several years of teaching experience), and are important and useful for students and professionals alike. This paper presents several problem transformation examples. These are expected to enhance the effectiveness of algorithmic thinking, problem-solving, and algorithm development.

1 Introduction

Algorithms are at the heart of all computing. In the increasingly information and computation driven world, a good grasp of algorithmic techniques is

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

extremely important for the development of elegant and efficient solutions to problems in various domains.

Also, algorithmic thinking is considered one of the 21st century skills and is important across several disciplines. Students and professionals in the Computing disciplines (Computer Science, Computer Engineering, Data Science, Information Systems, Artificial Intelligence, etc.) need to have algorithmic thinking and problem–solving skills for effectively solving information–rich, information–driven, real–world and societal problems. Even for technical professionals from other disciplines, it is beneficial to have a good grasp of algorithmic techniques, which they can apply effectively to solve problems in their chosen domains of specializations.

The term computational thinking was introduced in [6] where computational thinking has been described as consisting of a variety of elements drawn from Computer Science. Among other things, it involves solving problems, designing systems, reformulating a seemingly difficult problem into one we know how to solve, thinking recursively, using abstraction and decomposition, choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable, using heuristic reasoning to discover a solution, and several others. A variant, algorithmic thinking consists of key abilities that can be learned independently from programming. It is one of the required key abilities in Computing/Informatics. In [2], it has been shown that algorithmic thinking can be developed independently from learning programming, by the use of problems that may not be easy to solve, but have an easily understandable problem definition [2].

There have been several novel efforts in the development of techniques to facilitate and enhance algorithmic thinking. For example, there have been several examples of work in the areas of teaching computational/algorithmic thinking and programming without the use of computers, but by several paper–and–pencil methods and/or hands-on gadgets/tactile methods. A major aspect of the Australian Informatics Competition (AIC), which has a core focus on algorithms, is a pen–and–paper event, which is described in [1]. A new technique for implementing educational programming languages using tangible interface technology, which makes use of inexpensive and durable parts with no embedded electronics or power supplies, is described in [3]. Several hands–on gadgets (primarily made of wood / cardboard) which have been developed to facilitate novices in programming and computing to improve the comprehension of a few representative algorithms, and which serve as stepping stones to algorithmic thinking and programming are presented in [4, 5]. These gadgets are designed around several classical problems such as 0/1 knapsack problem, 2D packing problem, sorting, towers of Hanoi problem, etc. These gadgets also act as motivations for persons who may not have mathematical or computing

background, but are interested in problem solving and developing algorithms. Informal qualitative feedback has shown the effectiveness of the hands–on gadgets for the beginners, in facilitating a good understanding of the problems, and systematic thinking in solving the problems.

Based on several years of teaching algorithm design courses, and having experimented with several techniques to enhance the effectiveness of teaching and learning algorithms, we have identified some techniques which have been found to be effective in algorithmic thinking. One of the main themes of all of these techniques is the use of 'non–linearity' during teaching and learning. These provide opportunities for thinking in multiple ways and facilitate deeper understanding of the problem and the solution technique. Note that the term 'non–linearity' is used in an informal sense. It is meant to highlight the use of techniques, not commonly used in traditional lectures where straight–forward delivery of content with concepts and examples is the norm. These techniques facilitate viewing problems and thinking about solutions from different angles, and to make connections between seemingly different problems, that share a common solution structure.

Some of the components of the non–linear mode of teaching/content delivery are: (a) development of interesting/useful questions from data/facts; (b) development of counterexamples; (c) making connections between problems with similar 'computational' structures; (d) problem transformation to make use of existing computational elements/gadgets; etc. In this paper, we only focus on the last technique – making use of predefined (pre-existing) computational elements/gadgets, along with appropriate transformation of the problems.

2 Problem Transformation

Problem transformation has been a well–known technique in engineering which makes the understanding of the problem, and the solution development much easier / efficient than in the original domain of the problem. There are numerous transformation techniques. Just to give a couple of examples, Laplace transformation (from the time domain to the frequency domain) transforms differential equations into algebraic equations, and convolutions into multiplications. In signal processing, Discrete Fourier Transform (DFT) is used to transform time domain signal to frequency domain, which makes several operations, including noise removal, easier. Discrete Cosine Transform (DCT) is used to transform grey scale / colors of the pixels to (spatial) frequencies in image compression. In the multiplication of (large) polynomials, the coefficient vector is transformed to point-value representation using Fast Fourier Transform (FFT), which simplifies the computation enormously.

In Engineering and Computing disciplines, solutions to numerous problems have been developed and are continuously improved. For a given problem, instead of solving it from scratch, a better option would be to identify if solution to this or similar problem is already available. In many practical scenarios, transforming a given problem to a similar problem for which (an elegant) solution is already available would be beneficial in terms of time, effort, and cost savings. Thus, it is worthwhile to evaluate the feasibility of transforming a problem in order to make use of existing solution to the transformed problem, rather than jumping into development of solution from scratch.

2.1 Building algorithms using predefined computing blocks

One of the key skills in 'building' algorithms for several applications is not to start from scratch, but to use known and well–tested, pre-existing gadgets / computing elements / components / blocks, consisting of function, subprogram, software module, design pattern, or even specialized hardware (any combinations of them) in meaningful ways to obtain the solution to the problem. This requires (a) skills in transforming the original problem in such a way that the pre-existing computing elements could be used, and (b) for choosing the right set of blocks and their combinations. This is somewhat in spirit to the use of design patterns in software engineering. A design pattern provides a general reusable solution for the common problems occurring in software design. The patterns typically show relationships and interactions between classes or objects. Their use speeds up the development process by the use of well tested, proven development/design paradigm.

In order to use pre-existing computation elements, the problem needs to be transformed such that it could be given as the expected input to the pre-existing computation element(s). The solution produced by the pre-exiting computation element is subjected to an inverse transformation to obtain the solution to the original problem. An overview of the process is shown in Figure 1.

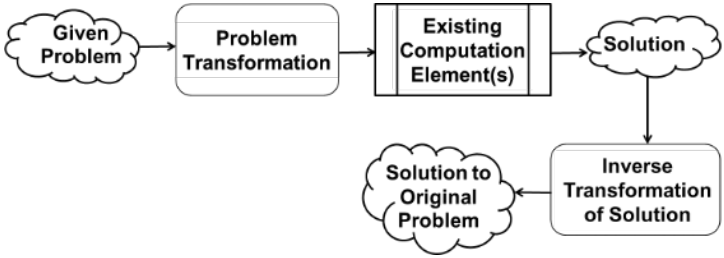


Figure 1: Problem transformation in order to use existing computation

This approach saves tremendous amounts of time, effort, and cost, since the solution is not developed from scratch, rather is composed of using already existing computing blocks. The cost of transformation and using the pre-existing computation element must be lower than the cost of solving the problem from scratch, which is usually the case in many cases.

3 Examples of use of predefined computing blocks

In this section, we present descriptions of a few problems, the descriptions of the computing blocks available, and the task of problem transformation and using/combining the available computing blocks to solve the given problem.

3.1 Interval overlap detection

The following gadgets / "computation units" in Figure 2 are already available. The "Inside Interval" gadget takes a point and an interval as inputs and outputs a Yes (No) depending on whether the point is inside the interval (or not). The AND and OR blocks perform the standard logical operations. An interval is a tuple (Si, Fi), where Si and Fi are real numbers (Si $<$ Fi,) representing the starting and ending points of the interval (temporal or spatial), and a point is a real number.

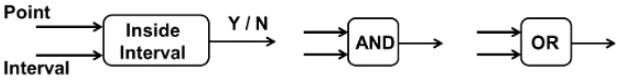


Figure 2: Gadgets for checking point containment in interval and elementary logic operations

Using only these existing (predefined) components, it is required to solve the interval overlap problem, i.e., to determine whether two intervals (Sk, Fk) and (SL, FL) overlap or not. One may use multiple copies of the gadgets, but it is desirable to use the minimum number possible.

It must be observed that for two intervals to overlap, at least one of the end points of one interval (say K) must be contained inside the other interval (say L). This logic is 'implemented' using combinations of the above gadgets (Figure 3).

3.2 Use of a "partition" gadget

Suppose a gadget (Figure 4) is available which takes as input any array A[L..R] of numbers, and partitions A and returns the index (new position of) 's' of P,

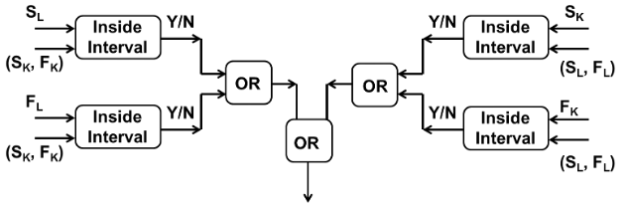


Figure 3: Using pre–existing gadgets for checking interval overlap

the pivot element A[L]. Note that after partition, all elements to the left of index 's' are less than A[L] and all the elements to the right of 's' are greater than or equal to A[L].

Figure 4: Partition gadget

Finding the number of elements smaller than a given value. Given A[0 .. N], an array of size N + 1 containing N numbers in A[1] through A[N], and a number (key) Q, it is required to find the number of elements of A which are >= Q, using the gadget of Figure 4 just once.

The leftmost element of A, A[L] is set to key Q: A[0] = Q. This array is then given as input to the "partition" gadget. The number of elements >= Q is easily seen to be (N – s).

Finding the smallest element. Suppose an array A[0..N–1] is given, and it is required to find the smallest number in A, using only the gadget of Figure 4.

It is easy to see that, if after the use of the gadget on an input A, the value of 's' is 0 or 1, then, A[0] must be the smallest element. If not, then use the gadget repeatedly on part of the array A[0.. s – 1], until s = 0 or 1, and return A[0]. This sequence of operations is formally stated in the following sequence of statements, where the repeated use of the gadget is given by the 'while' loop.

```
s = GADGET (A[L..R]);
while (s != 0 OR s != 1) do
  R = s - 1;
  s = GADGET (A[L..R]);
endwhile;
return A[L];
```

3.3 Use of a gadget that locates the minimum and maximum in an array

Suppose two gadgets (Figure **??**) are available – one that takes as input any array A[L..R] of numbers, and locates and returns the indices i and j of the minimum and maximum elements in the array, and the other swaps the contents of two memory locations given as inputs. Using only these gadgets, it is required to sort a given array A[0..N–1] in non–decreasing order.

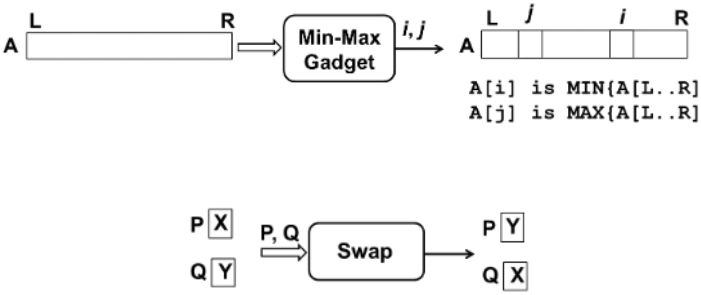


Figure 5: Gadget for locating minimum and maximum in an array, and gadget for swapping

By using the gadget once on an input array A[L..R], it is clear that A[i] and A[j] would be the minimum and maximum values among A[L] to A[R]. Then, the pairs (A[i], A[L]) and (A[j], A[R]) are swapped using the "Swap" gadget. After this, A[L] and A[R] will have the smallest and largest of the values among A[L] to A[R]. The gadget is next used on A[L+1 .. R–1] followed by swapping, and this is repeated as long as L < R. When the process terminates, the array A will be sorted. The repeated use of the gadgets is formally stated in the following sequence of statements.

```
while (L < R) do
   (i,j)  = GADGET(A[L..R]);
   Swap (A[L], A[i]);
   Swap (A[R], A[j]);
   L = L + 1; R = R - 1;
endwhile
```

3.4 Use of a gadget producing three partitions

The gadget (Figure 6) takes as input any array A[L..R] of numbers, and produces three partitions. It locates and returns the indices P and Q of A such

that (a) all elements to the left of P are < X, (b) all elements A[P] to A[Q] (inclusive) are equal to X, and (c) all elements to the right of Q are < X (where X = A[L], and is referred to as the pivot). Using this gadget repeatedly, it is required to sort a given array A[0 .. N–1] in non–decreasing order.

Figure 6: Gadget that produces three partitions

Note that after using the gadget once, the elements A[P] to A[Q] (inclusive) are in their final positions in the final sorted array. Subsequently, the gadget needs to be applied to the two partitions A[L .. P–1] and A[Q+1 .. R], repeatedly as long as P > L and Q < R. The repeated use of this gadget as described is captured in the following sequence of statements.

```
SORT_WITH_GADGET (A[L..R])
begin
  if (L == R)return;
  (P, Q) = Partition (A[L..R]);
  SORT_WITH_GADGET (A[L..P-1]);
  SORT_WITH_GADGET (A[Q+1..R]);
end
```

3.5 Use of a gadget that reverses part of a given array

Suppose a gadget or 'computation box' (Figure 7) is available, which takes as input an array A and two indices i and j (i <= j) and reverses the elements of A in the range i to j. The task is to use the gadget and perform left rotation by K positions of a given input array A[1 .. N].

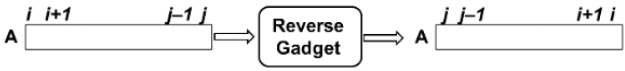


Figure 7: Gadget which reverse the input

Note that after rotation by left by K positions, the first element of the original array will be at index (N-K+1) MOD N, the second element of the original array at index (N-K+2) MOD N, etc. To accomplish this, the gadget can be used a minimum of three times: (i) first reverse the elements in the index range (1, K), (ii) then reverse the elements in the index range (K+1, N), and (iii) lastly, reverse the elements in the range (1, N). This can easily

be verified by writing out the indices after each step. The following sequence of gadget use with the indices shown will accomplish the task. The following sequence of gadget use with the indices shown will accomplish the task.

```
Reverse (A[1..K]);
Reverse (A[K+1..N]);
Reverse (A[1..N]);
```

3.6 Use of a gadget that computes the maximum matching in a given graph

Suppose a gadget 'computation box' (Figure 8) is available, which takes a connected graph as input, then computes and outputs the maximum matching in the graph. Note that a matching in a graph G = (V, E) is a set of all edges such that no two edges in the set are incident on a common vertex.

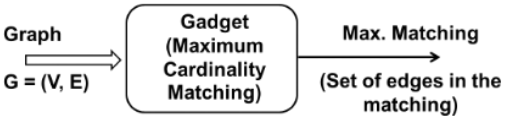


Figure 8: Gadget for computing the maximum matching in a given graph

Domino tiling. Given an N x N 'crossword grid' (Figure 9), it is required to determine if it can be tiled with at least K (1 x 2 sized) dominos, using only the gadget of Figure 8.

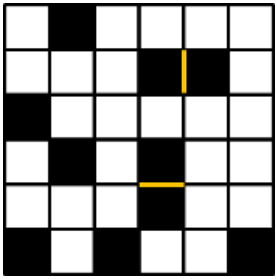


Figure 9: Crossword grid

Transform the grid into a graph where each empty cell is a vertex, and an edge between two adjacent empty cells. Supply this graph as input to the gadget to find the maximum matching, and check to see if K <= |M|, where M is Gadget's output, which is the maximum matching (set of edges).

International relief effort. An international relief effort is being organized for a disaster struck area. Each person can speak more than one language. It is required to form the maximum number of 2–person teams from out of N applicants such that every pair in every teams speaks a common language. This needs to be determined using only the gadget of Figure 8.

Build the graph G where the vertices are the persons. Add an edge between vertices i and j if they speak a common language. The problem then becomes one of finding the maximum cardinality matching in this graph. Supply G as input to the gadget of Figure 8, and the output would be the required solution.

4 Conclusions

Algorithmic thinking and problem solving is extremely important for students and professionals in the Computing disciplines (Computer Science, Computer Engineering, Data Science, Information Systems, Artificial Intelligence, etc.). Problem transformation skills are in important part of algorithmic thinking. Problem transformation techniques are practical and save time, effort, and cost in many scenarios by making use of pre-existing computational units. This paper presented some representative examples of making use of existing computational elements/gadgets with appropriate problem transformation which facilitate enhancement of algorithmic thinking skills.

References

[1] B. A. Burton. Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4:3–14, 2010.

[2] G. Futschek. Algorithmic thinking: The key for understanding computer science. In *IProceedings of the 2nd International Conference on Informatics in Secondary Schools: Evolution and Perspectives*, pages 159–168, 2006.

[3] Michael S. Horn and Robert J. K. Jacob. Designing tangible programming languages for classroom use. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, page 159–162, New York, NY, USA, 2007. Association for Computing Machinery.

[4] S.R. Subramanya. Hands–on gadgets to facilitate algorithmic thinking for beginners. *International Journal of Advanced Research in Computer Science & Technology*, 2(2):521–526, 2014.

[5] S.R. Subramanya. Hands–on gadgets to facilitate and augment algorithmic thinking and problem solving for beginners. *International Journal of Software & Hardware Research in Engineering*, 4(6):42–50, 2016.

[6] J. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

A Bottom-Up Approach for Computer Programming Education*

Lasanthi N. Gamage
Math and Computer Science Department
Webster University
St. Louis, MO 63119
lasanthigamage67@webster.edu

Abstract

Introductory computer programming is a challenging course for many students, demonstrating higher withdrawal and failure rates. This article discusses a Bottom-up teaching approach that reverses the traditional teaching method in introduction-level computer programming (CS1). This approach consists of a series of in-class activities. An activity starts with a full computer program that demonstrates the definitive topics the instructor desires to introduce - instead of introducing detached multiple topics that they will be assembling later on. This article enumerates a sequence of in-class activities along with a few specific activities. The students' performance in CS1 class and next-level programming (CS2) shows this approach's effectiveness. Transformation into the set up does not require additional time nor resources from instructors, making this is scalable on the class capacity.

1 Introduction

Computer Programming in Computer Science Degree Programs is undoubtedly an essential course for the degree. On the other hand, most computer science undergraduates find programming to be one of the most challenging courses in their degree path due to rigid syntax, unfamiliar structure, and time spent to produce a simple output, for example [3].

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Due to this challenging nature of programming, lower-level programming courses experience higher withdrawal and failure rates, leading to lower retention rates [1]. The statistics show that the first-year students who withdraw from their classes are significantly less likely to persist to a degree [1]. The consequences of higher withdrawal and failure rates impact many parties of a university. Some of which include the students (waste of money and time), the department (upper-level course cancellations due to less enrollment), and eventually, the University (higher average time to graduate and lower student retention) [3, 13]. These consequences and the challenges have developed the scholars' research interest in Computer Programming Teaching and Learning.

One typical teaching method for computer programming, named the top-down approach in this article, introduces programming concepts one-by-one and then exercises several examples on the concept. Finally, writing a full program that exercises the topics. However, these full programs require many concepts to be taught to understand each piece of the program. Thus, it requires several topics to be covered and a significant amount of class meeting time before going for a full programming assignment.

The teaching technique proposed in this article is named the *bottom-up approach*. To make programming more enjoyable and understandable, it introduces bite-size topics in a full program. The full programs are incomplete, and the students will be completing the programs by filling in the blanks as they learn the new concepts; these activities will be called *in-class-activities (ICA's)*. The instructor would introduce each concept centering to the ICA. The instructor expands the discussion on specific topics with more examples and the topic's technical background accordingly.

With this bottom-up approach, the students get a broader view of the concept and its application with the full program first. Nevertheless, the students' concentration should only be on new topics. This approach helps students grasp the lesson's essence by separating other possible concerns/distractions, such as rigid syntax and unfamiliar structure, from students' minds, referred to in [3]. After presenting the concepts parallel to ICA's, complex take-home assignments will improve their logical thinking and analytical skills. As a summary: this work presents:

A Novel Pedagogical method: This method is different from the most popular techniques we see in the literature (presented in Section 2)

A list of example ICAs: Ten ICAs are presented with selected in detailed versions.

A Comparison of Student Performance: The student performance is presented that shows the influence of a strong foundation to be successful in subsequent courses.

The organization of the paper is as follows. The article continues with the

work related to teaching introductory level programming in Section 2. Section 3 discusses the procedure followed in this project. Section 4 discusses ten selected ICA's with some tangible examples. Section 5 presents the results and Section 6 discusses advantages and issues of this method and the future work.

2 Related Work

It is a common understanding that learning programming is challenging. Among the many reasons for its challenging nature, some of them include rigid syntax, unfamiliar structure, unsuitable learning styles, lack of motivation, and need for multiple skills and knowledge [3, 9]. The educators try out different strategies to make learning exciting, fun and to improve retention. This section addresses different strategies experimented with to achieve the objectives mentioned above. Some such strategies that teachers have adopted are introducing active learning approaches, teaching in computer labs, class time assignment, having mandatory lab class, flipped classroom, group assignments, 2-stage assignment submissions [5, 14, 2, 18, 6, 17]. Syeda et al. have studied the classroom setup (active learning classroom vs. traditional lecture hall) on student performance [15]. Some other experiments consider using other tools such as visual aided programming languages (e.g., Alice and Scratch [11]), game-based approaches [16], different applications (e.g., Mind mapping software [10]). In addition to the above strategies, some researchers introduce different modeling approaches [12] to teach programming, especially object-oriented concepts.

Active learning approaches would be useful in learning as, in most cases, the students get a chance to apply their critical thinking. However, it still needs instructors' supervision and guidance for struggling students, making it less practical for large class sizes. The proposed method's advantage is that it would scale up with the class size because the instructions with this method are straightforward, and no additional human resources are needed.

Alen et al. [4], and Dawar [7] have experimented with approaches similar to the one proposing in this article. They have named their approach Many Small Programs (MSP) and An Assignment A Day (AAAD), respectively. Both works see the advantage of breaking down the big-scale assignments into smaller ones. Like our motivation, they also believe that big-scale assignments may require students to complete several interconnected parts and tend to have lots of text to explain before the assignment is completed. Nevertheless, one significant difference in our proposed approach is that it uses these programs as in-class activities, whereas they use them in assignments. The simpleness and the bite-size granularity of the ICAs better fit in introducing the text rather than assignments. Another difference is that ICAs do not require heavy grading. In contrast, assignments in [4] and [7] do, because it is a teacher-

guided atmosphere from which students' proficiency should not be evaluated; the intention of these activities was not to assess them, in the first place. However, the students can be asked to submit ICAs for zero-or-all grading to promote students' active participation, which would not need much effort or time to grade.

3 Methodology

Class Setup

The project [1] explained here was piloted in two sections of introductory level Computer Programming classes (CS1's) in Fall 2019. The course covers the first seven chapters of [8]; the next stage programming course, CS2, covers the remaining chapters of the textbook; CS1 is offered three days a week, each for 50 mins, the class capacity is 20 students, and, in exceptional cases, it would go up to 25. Full-time or part-time faculty teach the classes, and no class-assistants are available for neither teaching nor grading. The classes are solely lecture-classes, and no lab class is associated [2]. In the remaining weeks, the students will be completing the homework assignments on their own.

3.1 Project Design

ICAs exercise different topics that the instructor plan to cover for the day. The number of ICA's could depend on the instructor. Each ICA is assembled into a handout which composes the activity number, objective, assignment, and, in most cases, an outline of the program (see Figure 1); note that this method does not expect any prior student preparation like in flipped classrooms.

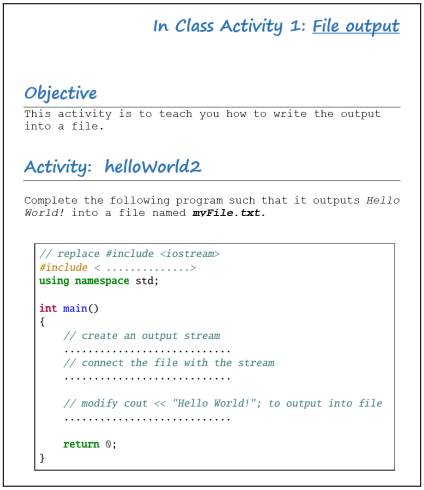
The outline helps in multiple ways. One is that students get a comprehensive picture of the application of the topic (although some topics have not been introduced yet) with the outline.

Student focus can be narrowed down to the new topics only (other known and unknown concepts are not focused on). Class time can be utilized for a productive conversation of the new topics (but not the other related concepts) and clear students' concerns, reinforce the previously known topics and de-emphasize the non-introduced topics.

The programs provided in these activities are intentionally made simple to make it easier to follow and reduce the assignment's information gathering

[1] This research was reviewed and determined to be exempt by the IRB at Webster University

[2] On the very first day, however, the students will be working on a HelloWorld program in a lab to make them familiarize with the institutional computer facilities and the IDE, for example.



In Class Activity 1: File output

Objective

This activity is to teach you how to write the output into a file.

Activity: helloWorld2

Complete the following program such that it outputs *Hello World!* into a file named **myFile.txt**.

```
// replace #include <iostream>
#include < ..............>
using namespace std;

int main()
{
    // create an output stream
    ..........................
    // connect the file with the stream
    ..........................

    // modify cout << "Hello World!"; to output into file
    ..........................

    return 0;
}
```

Figure 1: In Class Activity 1 (File output)

time. ICA's composed of two sub-activities: filling in blanks of a program and rewriting the whole program on a separate paper or typing it in Visual Studio or both. Each part has its learning outcomes. The first one is to teach them the topic-specific content, whereas the second sub-activity is to practice the concept's application in a full program. To enforce these learning objectives, optionally - but advisably-, the instructor could make students submit the second part of the activity for points (The authors accredited those points for class participation category, which is 5% of the final grade). The grading could be as lenient as zero-or-nothing grading because it is not necessarily required logical thinking. Those types of assignments include implementing a two-player tic-tac-toe game against the computer and generate a two-digit subtraction worksheet - 4 problems per line and six lines.

4 In-Class Activities

The first activity ICA1 introduces file output (Note that file output was introduced as early as the second week as opposed to the order in the textbook so that the student can practice it enough before they get to the Functions. If an instructor does not find this change necessary, he/she could move this activity to an appropriate place in the list). ICA1 adopts the typical $\text{helloWorld}_{\text{cout}}$ program and asks to complete $\text{helloWorld}_{\text{fout}}$ program in which directs the output to a file. Figure 1 illustrates the complete view of the activity (in the remaining samples of ICAs, we only present the program outline). Centering

on ICA1, the instructor can lead into a meaningful discussion by comparing the counterparts of the $\text{helloWorld}_{\text{cout}}$ and $\text{helloWorld}_{\text{fout}}$, such as **c++** iostream vs. **c++** fstream, **c++** cout vs. **c++** fout (the file object was named **fout** to maintain the comparableness), and the role of **c++** std vs. file stream object and file linking to the object.

ICA2 introduces variable definition, initialization, constant definition, and mathematical expression; since each concept is easy to digest, they fit fine in the same ICA.

The next two activities, ICA3 and ICA4, introduce *if-else* and *switch-case*, respectively. Both share the same assignment, which displays the day of the week in words based on the user given integer. As noted before, the assignment itself is simple, promoting the concentration on the topic. Even though ICA3 was the first time that the students experience branching, with ICA3, they experience multi-branched *if- else if* statements. This activity renovates the standard order of presenting the if branching statement to students. This topic, typically, approaches a step-by-step expansion of the *if*-statement, starting with only one branch and eventually reaches multi-branched *if- else if*. ICA3, instead, starts with multi-branched *if*- statement and then, teaches its variations - one branch (if branch) and two branches (both *if- else* and *if- else if*)- which is easier for students to understand.

The next exciting set of activities is on functions. A lesson plan with ICA's for functions would take a minimum of four in-class activities: reviewing system-defined function-calls (ICA5), introducing user-defined function-calls (ICA6), introducing function definitions (ICA7), and designing and writing a FULL program with functions (ICA8).

Although the students have experienced system-defined and function-calls, some students demonstrate a deficiency in their correspondence of user-defined functions. For example, some students are under the impression that the parameter and argument names should be the same even though they used system-defined functions with zero information about the parameter names. Another set of students cannot think about invoking a user-defined function several times to get a similar effect with different arguments. They have done with system-defined functions very often. ICA5 is to clear these common misconceptions using system-defined functions.

ICA6 lets students practice function calls furthermore. This activity employs user-defined functions. It provides prototypes of all the functions required (see Figure 2), and the students will be writing the function calls to get the output shown in Figure 2. ICA6 is an excellent position to emphasize the role of include libraries used in ICA5, such as **c++** **#include** <**cmath**>, vs. the prototypes in ICA6. It can again be reinforced in ICA7 in where the students will be practicing function definitions.

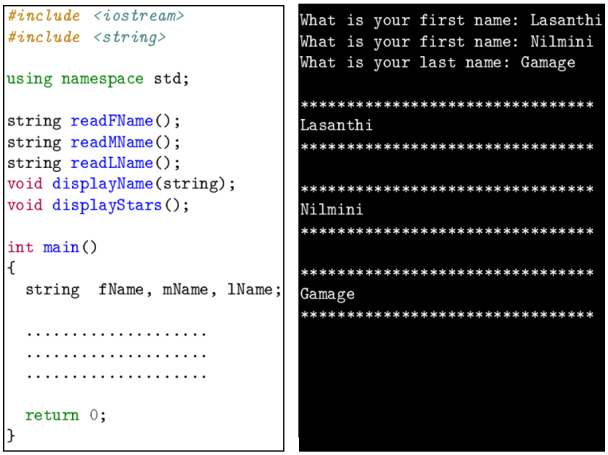


Figure 2: In Class Activity 6 (User-defined Function Calls) and It's Sample Output

ICA7 is an extension of ICA6. It focuses on function-definitions. It uses the same programming problem as it was in ICA6. It, however, does provide a second opportunity to revise function-prototypes and function invoke while introducing definitions.

ICA8 is also on functions, and in this activity, the students get to work on a different problem to apply all three aspects of functions from scratch. This activity does not teach any new concepts; instead, it gives them a chance to practice what they learned in three preceding activities.

ICA9 and ICA10 are on arrays: one and two dimensional, respectively (Yet, authors used more ICA's for arrays which were not reported here due to the space limitations, for example, ICA's on partially filled arrays, parallel arrays, and arrays in functions.). ICA9 and ICA10 demonstrate the array definition, size declarator, initialization, and traversing through the array in the selected type of dimension.

5 Results

Here we present the results in terms of the student grades. In this research, we first report the CS1 grades of the students who experienced the proposed method. That is two sections of Fall 2019 - 16 and 7 students taught by the same instructor (we refer this group of students G_{CS1}). The control group contains the students from the other CS1 sections taught by other instructors. The instructors teaching methods were not influenced by any means, and the

corresponding instructors independently determined those. Figure 3 (a) shows the grade percentage of the G_{CS1} and according to that 52% of the students have got an A grade and 65% of them (A and B grades) move to CS2.

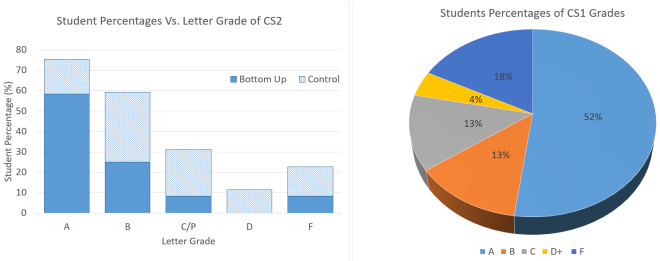


Figure 3: Students CS1 and CS2 grades

We next present the performance of those who took CS2 in Spring 19. We present the grades of those students in two groups: those who underwent the Bottom-up Approach $CS2_A$ and those who were not ($CS2_B$). It is worth noting here that $CS2_A$ is a subset of G_{CS1}; That is because CS2 is taken only by those who have a B- or above in CS1, and not all students take CS2 immediate next semester. Same with ($CS2_B$), meaning that the semester those students took CS1 could spread into several semesters into the past.

As Figure 3 (b) illustrates, close to 60% and 20% students from $CS2_A$ and $CS2_B$, respectively, have obtained an A grade in CS2. Furthermore, more than 90% of $CS2_A$ have got a Pass grade for CS2 whereas that of $CS2_B$ is only 74%. Also, only 51% of $CS2_B$ have satisfied the prerequisite for the Data Structures(DS) course, where as 83% of $CS2_A$ have satisfied the DS prerequisites. These results show that even with the same treatment in CS2, the influence of a solid foundation in CS1 has lead $CS2_A$ to have better grades.

6 Discussion

preparation time:: The ICA's take a moderate amount of preparation time; they would not take more time than assignments or tests. One reason for that is that the assignments in activities could be quite naive, requiring less contemplation and time. This kind of relaxed preparation is acceptable for the activities because the main objective of activities is to demonstrate the topics and not assess their problem-solving capabilities. Another reason is that the instructors do not need to develop unique assignments from semester to semester to maintain academic integrity; instead, the instructor could repeatedly use the same set of activities.

grading: As it was pointed out at a couple of places before, grading these activities is optional. If an instructor decides to grade to promote students' active participation, it could be as relaxed as zero-or-nothing grading. This grading would not require additional grading tools or supplementary grading assistants.

scale-up: The presented pedagogical method scales up well with the class size. The primary reason for that is that it is merely a part of the class plan, requiring no one-on-one teacher attention. Each activity has clear directions for the students in the handout, and it only expects students to follow the instructor-led discussion. Thus, these activities are easy to implement in both small and large-scale class environments. Also, since ICA's could solely be paper-based, it does not require a change in the venue, such as from a lab to lecture- hall or vice versa. Due to the above reasons, the presented method fits well with any class-formats.

Issues: One possible issue an educator might face and resolve in this method is that figuring out the suitable granularity of concepts that one should be putting in one activity. The educator can decide the right granularity as the semester goes by based on the student population. Alternatively, the instructor could use multiple activities one day if the assignments were designed with finer granularity.

Future Work: The Authors expect to expand this project in a larger class set up as a joint work with a different institute.

References

[1] Clifford Adelman. *The Toolbox Revisited: Paths to Degree Completion From High School Through College*. U.S. Department of Education, 02 2006.

[2] Gokce Akcayir and Murat Akçayır. The flipped classroom: A review of its advantages and challenges. *Computers & Education*, 126, 08 2018.

[3] Azad Ali and Charles Shubra. Efforts to reverse the trend of enrollment decline in computer science programs. *Issues in Informing Science and Information Technology*, 7:209–224, 2010.

[4] Joe Allen, Frank Vahid, Alex Edgcomb, Kelly Downey, and Kris Miller. An analysis of using many small programs in cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 585–591, 02 2019.

[5] Dhruva Chakravorty, Marinus Pennings, Honggao Liu, Zengyu Wei, Dylan Rodriguez, Levi Jordan, Donald McMullen, Noushin Ghaffari, Shaina Le, Derek Rodriquez, Crystal Buchanan, and Nathan Gober. Evaluating active learning approaches for teaching intermediate programming at an early undergraduate level. *The Journal of Computational Science Education*, 10:61–66, 01 2019.

[6] Juan Chen, Yingjun Cao, Linlin Du, Youwen Ouyang, and Li Shen. Improve student performance using moderated two-stage projects. In *CompEd '19: Proceedings of the ACM Conference on Global Computing Education*, pages 201–207, 04 2019.

[7] Deepak Dawar and Marianne Murphy. An assignment a day scaffolded learning approach for teaching introductory computer programming. In *Proceedings of the Information Systems Education Journal*, volume 18, pages 59–73, 2020.

[8] Tony Gaddis, Judy Walters, and Godfrey Muganda. *Starting Out with C++: Early Objects*. Addison-Wesley Publishing Company, USA, 7th edition, 2010.

[9] Tony Jenkins. On the Difficulty of Learning to Program. In *Loughborough University*, 2002.

[10] Yizhen Liu, Yingxin Tong, and Yuqi Yang. The application of mind mapping into college computer programming teaching. *Procedia Computer Science*, 129:66–70, 01 2018.

[11] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10:16, 11 2010.

[12] Michela Pedroni and Bertrand Meyer. Object-oriented modeling of object-oriented concepts. In *Teaching Fundamentals Concepts of Informatics. ISSEP 2010. Lecture Notes in Computer Science*, pages 155–169, 01 2010.

[13] Markeya Peteranetz and Leen-Kiat Soh. A multi-level analysis of the relationship between instructional practices and retention in computer science. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 37–43, 02 2020.

[14] Yinghui Shi, Yanqiong Ma, Jason MacLeod, and Harrison Hao Yang. College students' cognitive learning outcomes in flipped classroom instruction: a meta-analysis of the empirical literature. *Journal of Computers in Education*, 7, 05 2019.

[15] Ayesha Syeda, Rutwa Engineer, and Bogdan Simion. Analyzing the effects of active learning classrooms in cs2. In *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 93–99, 02 2020.

[16] Yoke Seng Wong and Maizatul Yatim. Computer game as learning and teaching tool for object oriented programming in higher education institution. *Procedia - Social and Behavioral Sciences*, 123, 03 2014.

[17] Hans Yuan and Yingjun Cao. Hybrid pair programming - a promising alternative to standard pair programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1046–1052, 02 2019.

[18] Shanna Zhuang, Hui Wang, WenBin Zhao, Tongrang Fan, and Yumei Zhang. Practical guidance method for c/c++ teaching reform. In *2015 10th International Conference on Computer Science & Education (ICCSE)*, pages 834–837, 07 2015.

Making Change: Rigorous Software Construction as Experiential Learning*

Michael Kart
Department of Computer Sciences
Saint Edward's University
Austin, TX 78704
michaelkart@stedwards.edu

Abstract

In his proposal "Constructionism: A New Opportunity for Elementary Science Education," Seymour Papert starts with the constructivist theory viewpoint that learning is building knowledge rather than receiving a transmission of knowledge. He then defines constructionism as an extension of this theory which includes the learner constructing a meaningful product. This paper investigates the effect of applying this viewpoint to the teaching of computer science content via the rigorous construction of a related software artifact. Specifically, radix representation concepts and algorithms are taught using the construction of a change making class. The results show that, with appropriate support, software construction can greatly enhance student learning in computer science.

1 Introduction

Computer science students have to a lot to learn. Seymour Papert [3, 4] contended that the act of constructing artifacts can produce deep learning opportunities in his theory of "constructionism." diSessa argues that programming can facilitate learning since the act of programming forces the programmer to represent their solution in a structured way while providing opportunities for them to reflect on their thinking [2].

*Copyright ©2021 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

This paper asks the question: "Can the act of software construction combined with prior knowledge, rigor, and experiential learning result in certain indirectly related learning outcomes?" In particular, can the universal knowledge of making change (e.g., the change for 98¢ is 3 quarters, 2 dimes, and 3 pennies) along with rigorous specification (e.g., preconditions and postconditions, and JUnit test cases) and implementation be leveraged to affect the learning outcomes for radix representations. All of this takes place in a third programming course while using experiential learning techniques.

The course proceeded as follows:

1. Instructor introduced the concept of making change and ensured that all students realized that they already knew how to do it
2. Instructor introduced the instructor-authored ChangeMaker interface; students helped complete the interface by determining preconditions and postconditions
3. Instructor distributed several instructor-authored concrete examples of ChangeMaker behavior; students had to figure out behavior for fifteen more examples
4. Instructor distributed several instructor-authored JUnit test cases; students had to write fifteen more JUnit test cases
5. Students implemented the ChangeMaker interface
6. Instructor introduced the connections between ChangeMaker and radix representations
7. Instructor assessed radix representation learning outcomes on the final exam

2 Background

2.1 Radix Representations

Let n be a nonnegative integer and b be an integer that is greater than 1. Then, for some integer $k \geq 0$, n can be uniquely decomposed over the nonnegative powers of b as follows:

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0, \tag{1}$$

where $0 < r_k < b$ and $0 \leq r_i < b$, for each $i \in [0, k)$. The numeral $r'_k r'_{k-1} r'_{k-2} \cdots r'_0$ is defined to be the representation of n in radix (i.e., base) b, where r'_i is the digit that represents the integer r_i, for each $i \in [0, k)$.

For example, with $n = 98$ and $b = 2$:

$$98 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0, \tag{2}$$

Therefore, the radix 2 representation of 98 is 1100010.

There are two commonly taught algorithms for determining the base-b representation of $n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0$. The traditional algorithm determines the coefficients in the order r_0 up to r_k, while the alternative algorithm determines the coefficients in the order r_k down to r_0.

2.2 Traditional Algorithm and Perspective

The traditional way to teach how to obtain the radix representation of a particular number often involves the following presentation. Let n be a nonnegative integer and let b be an integer that is greater than 1. Assume that

$$n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0, \tag{3}$$

where $0 < r_k < b$ and $0 \leq r_i < b$, for each $i \in [0, k)$.

The traditional algorithm is based on the following observations:

$$n\%b = (r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0)\%b = r_0 \tag{4}$$

and

$$n/b = (r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0)/b \tag{5}$$

$$= r_k \cdot b^{k-1} + r_{k-1} \cdot b^{k-2} + r_{k-2} \cdot b^{k-3} + \cdots + r_1 \cdot b^0, \tag{6}$$

where '/' above denotes integer division.

So, the following algorithm produces the sequence of integers $r_k, r_{k-1}, r_{k-2}, \cdots, r_0$ (and, consequently, determines the sequence of digits $r'_k, r'_{k-1}, r'_{k-2}, \cdots, r'_0$):

```
public static List<Integer> getCoefficients(int n, int b)
{
        assert n >= 0 : "n = " + n + " < 0!";
        assert b > 1 : "b = " + b + " <= 1!";
        List<Integer> coefficients = new ArrayList<Integer>();
        if(n == 0)      //special case
        {
                int r_0 = 0;
                coefficients.add(r_0);
        }

        int n_prefix = n;
```

```
        int i = 0;      //for documentation purposes only
        boolean existMorePositiveCoefficients = (n_prefix > 0);
        while(existMorePositiveCoefficients)
        {
                int r_i = n_prefix % b;
                coefficients.add(0, r_i);
                n_prefix = n_prefix/b;
                existMorePositiveCoefficients = (n_prefix > 0);
                if(existMorePositiveCoefficients) i++;
        }
        return coefficients;
}
```

Most computer science students find this algorithm very opaque and pure memorization is their only chance of success. Moreover, this success is usually short-lived.

3 Alternative Algorithm and New Perspective

The alternative algorithm is the one that determines the base-b coefficients of $n = r_k \cdot b^k + r_{k-1} \cdot b^{k-1} + r_{k-2} \cdot b^{k-2} + \cdots + r_0 \cdot b^0$ in the order from r_k down to r_0. As mentioned earlier, an important aspect is the new perspective, which helps students gain and retain a better understanding of radix representations and their corresponding algorithms. This new perspective requires the introduction of the concept of a mixed-radix representation.

3.0.1 Mixed-Radix Representations

Let n be a nonnegative integer and let $[d_i : d_i > 1]$ be a (potentially finite) list of integers. Suppose that n can be decomposed over the d_i, for some integer $k \geq 0$, as follows:

$$n = r_k \cdot d_k + r_{k-1} \cdot d_{k-1} + r_{k-2} \cdot d_{k-2} + \cdots + r_0 \cdot d_0, \tag{7}$$

The numeral $r'_k r'_{k-1} r'_{k-2} \cdots r'_0$ is defined to be a representation of n in mixed-radix $[d_i]$, where r'_i is the digit that represents r_i, for each $i \in [0, k)$.

For example, when $n = 98$ and $[d_i] = [25, 10, 5, 1]$, n can be decomposed as follows:

$$98 = 3 \cdot 25 + 2 \cdot 10 + 0 \cdot 5 + 3 \cdot 1 \tag{8}$$

Therefore, 3203 is a representation of 97 in the mixed radix $[25, 10, 5, 1]$.

Notice that since $98 = 2 \cdot 25 + 4 \cdot 10 + 1 \cdot 5 + 3 \cdot 1$, 2413 also represents 98 and mixed-radix representations are not guaranteed to be unique. Moreover, notice that $n = 3$ has no mixed-radix representation over $[7, 4]$, so mixed-radix representations are not guaranteed to exist.

3.1 ChangeMaker

A ChangeMaker is an abstraction of a ballpark vendor who, after receiving $20 in payment for an $12.50 order, must return $7.50 in change, using coins, to the customer. Each ChangeMaker instance has a finite set of coin types, but an infinite number of coins of each type. In addition, a ChangeMaker must always make change in a *greedy* manner. This will be defined precisely below, but, for now, it can be taken to mean "do not give back two nickels when you could give back a dime instead."

3.1.1 ChangeMaker Interface

```
public interface ChangeMaker {
        //part of post:
        //for i in [0, rv.size() - 1): rv.get(i) > rv.get(i + 1)
        public List<Integer> getDenominations();

        //precondition: left to student
        //postcondition: left to student
        public boolean canMakeExactChange(int valueInCents);

        //precondition: left to student
        //postcondition: left to student
        //part of pre: canMakeExactChange(valueInCents)
        //part of post: valueOfChangeList(rv) == valueInCents
        public List<Integer> getExactChange(int valueInCents);

        //precondition: left to student
        //postcondition: left to student
        //part of pre:
        //changeList.size() == getDenominations().size()
        public int valueOfChangeList(List<Integer> changeList);
}
```

Notice that the interface doesn't have involve the concept of coin, instead, this has been abstracted up to the concept of denomination. The greedy constraint is manifested in the postcondition of the getExactChange() method. To make the expression of this constraint easier, let $[d_0, d_1, \cdots, d_k]$ be the return value from the method call getDenominations() and let $[c_0, c_1, \cdots, c_k]$ be the return value from the method call getExactChange(n). Then, the greedy constraint can be expressed as:

$$d_i > c_{i+1} \cdot d_{i+1}, \text{for each } i \in \{0, 1, \cdots, k-1\} \tag{9}$$

For example, suppose that m is a ChangeMaker instance such that:

$$\text{m.getDenominations()} = [100, 25, 10, 5, 1] = [d_0, d_1, d_2, d_3, d_4].$$

Then the method call m.getExactChange(98) returns [0, 3, 2, 0, 3] and the method call m.valueOfChangeList([0, 3, 2, 0, 3]) returns 98. Notice that the method call m.valueOfChangeList([0, 2, 4, 1, 3]) also returns 98, even though the list [0, 2, 4, 1, 3] = $[c'_0, c'_1, c'_2, c'_3, c'_4]$ violates the greedy constraint above since

$$25 = d_1 \leq c'_2 \cdot d_2 = 4 \cdot 10 = 40 \tag{10}$$

4 Greedy Change Making Algorithm

From experience, every student already knows the correct behavior of the method getExactChange(). For instance, all students can determine that the correct *greedy* change for 98¢ when the denomination list is $[100, 25, 10, 5, 1]$. Note that the following four inequalities are true:

$$3 \cdot 25 <= 98 < 4 \cdot 25 \tag{11}$$

$$2 \cdot 10 <= 23 < 3 \cdot 10 \tag{12}$$

$$0 \cdot 5 <= 3 < 1 \cdot 5 \tag{13}$$

$$3 \cdot 1 <= 3 < 4 \cdot 1 \tag{14}$$

So, greedy change for 98¢ involves exactly 3 quarters, 2 dimes, 0 nickels, and 3 pennies.

4.1 Connection to Radix Representations

Notice that the problem of making greedy change over a generic denomination list is equivalent to the problem of determining the coefficients in the expansion of Equation (7) for a mixed radix of $[d_i]$, where this list is strictly decreasing. In particular, for any $b > 1$, the problem of making greedy change over the denominations $[b^k, b^{k-1}, b^{k-2}, \cdots, b^0]$, is equivalent to the problem of determining the coefficients in the expansion of Equation (1).

For example, consider the following steps to calculate the coefficients involved in the binary representation of the integer 19. First, construct an instance of ChangeMaker such that the denomination list is $[2^5, 2^4, 2^3, 2^2, 2^1, 2^0]$. Second, make the method call getExactChange(19). This will return the change list $[0, 1, 0, 0, 1, 1]$, which enumerates the exact coefficients in (1) since $19 = 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$.

5 Methodology

Students learn about radix representations and associated algorithms via various active learning steps. First, students work through straightforward concrete change making problem over the "standard U.S. coins" involving Sacagaweas, quarters, dimes, nickels and pennies, that is, a denomination list of $[100¢, 25¢, 10¢, 5¢, 1¢]$. An example of this work includes: Assuming the standard U.S. coins, what change should a cashier give back to a customer who is owed 98¢? Typically, students notice that the $[0, 3, 2, 0, 1]$ case is special among all change lists with this value and at least one student brings up the $[0, 0, 0, 0, 98]$ case. At this point, the greedy constraint is organically introduced and discussed.

Next, students encounter a couple of less straightforward questions, while still using the standard U.S. coins. For instance, the question of whether change can be made for 0¢ is discussed. The instructor raises this question if no student does. The instructor then guides the class to the consensus that the correct answer is $[0, 0, 0, 0, 0]$. Another question includes whether greedy change can be make for $-11¢$. Students usually propose one of two candidates:

- No, it cannot.
- The change list is $[0, 0, -1, 0, -1]$

There is always active discussion about the $[0, 0, -1, 0, -1]$ change list candidate and how such a change list should be interpreted. The instructor guides the class to consensus that the correct answer is that no change can be made in this case.

Next, students work on concrete problems (provided by the instructor) involving denomination lists that differ from the U.S. coins. One interesting problem involves determining whether change can be made for 6¢ over the denomination list $[5, 3] = [d_0, d_1]$. The nuanced, and often surprising, answer is "No" since the change list of $[0, 2] = [c_0, c_1]$ violates the greedy constraint because $5 = d_0 \leq c_1 \cdot d_1 = 2 \cdot 3 = 6$ (see Inequality (9) above). In addition, test cases involving the denomination lists of $[2^8, 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0]$ ("truncated binary") and $[8^3, 8^2, 8^1, 8^0]$ ("truncated octal") are introduced; hexadecimal is held in reserve.

At this point, students receive the ChangeMaker interface and several JUnit test cases to the class. An example test case is shown in the next section.

5.1 ChangeMaker Test Cases

```
public void usChangeFor65Cents()
{
        //{100, 25, 10, 5, 1}
        Set<Integer> usDenomSet = getUSDenominationSet();
        ChangeMaker cm_STUDENT = getChangeMaker(usDenomSet);
        int valueInCents = 65;

        boolean actualCanMakeChange =
                cm_STUDENT.canMakeChange(valueInCents);
        assertTrue("canMakeChange(" + valueInCents + ")
                should be true!", actualCanMakeChange);

        List<Integer> actualChangeList =
                cm_STUDENT.getExactChange(valueInCents);
        List<Integer> correctChangeList =
                Arrays.asList(new Integer[]{0, 2, 1, 1, 0});
        assertEquals("Calculated changeList disagrees with
                expected!", correctChangeList, actualChangeList);
}
```

A full class period is dedicated to having each student actively generate the 20 JUnit test cases, which correspond in a one-to-one fashion to their previously developed concrete examples. Note that this is *purely* an activity in expressing these concrete examples as JUnit test cases. Some students finish within the hour and while others finish outside of class time.

At this point, the students all understand the concept of greedy change making along with its associated algorithms and most students have a full set of correct and diverse JUnit test cases. Only after students had achieved a conceptual understanding of ChangeMaker and were fully equipped with JUnit test cases were they tasked with the programming assignment that involves implementing the ChangeMaker interface.

Then, once students finish the programming assignment, they learn that they already know how to determine the radix representation for any base; especially base-2, base-8, and, after adding extra digits so that $r_i = 10 \implies r'_i =$ 'A', $r_i = 11 \implies r'_i =$ 'B', $\ldots$, $r_i = 15 \implies r'_i =$ 'F', base-16.

Lastly, the instructor assessed how well students understand radix representations and their associated algorithms on the final exam.

6 Discussion

Featuring greedy change making in a sophomore-level programming class has many benefits. First and foremost, as mentioned above, *everyone already understands how to make change*, so there is not a single student who is confused

about any of the straightforward concrete examples. Students can then focus on the *expression* of these concrete examples in terms of method calls and JUnit test cases. Furthermore, students acquire the ability to self-author a high-confidence set of JUnit test cases *before* they begin programming. These aspects are extremely important to enable a greater amount of active learning for the rest of the course and their academic career as well. Secondly, the ChangeMaker concept has just enough complexity and nuance to make a memorization-based approach unlikely to succeed. The ChangeMaker assignment, paired with concrete examples, and JUnit test cases allow these students to try such an approach and have the chance to see (via failing test cases) *that they need a different approach.* As Boaler [1] points out, students who rely on memorization are the lowest achieving (math) students in the world.

In the course, the difference between the change lists produced by a "hexadecimal" ChangeMaker (i.e., all of the denominations are powers of 16) and the corresponding hexadecimal representation is emphasized. That is, the change list $[15, 2, 3, 12]$ is different from "F23C". From a data type perspective, students were best served when the hexadecimal representation is typed as a String and they were able to see that the change list is not the hexadecimal representation, but, instead, is highly correlated with it. So, students also got to wrestle with the difference between a value (e.g., r_k) and the presentation of that value (e.g., r'_k).

ChangeMaker can be used to implement other unrelated (in the eyes of a sophomore) concepts, such as certain calendars, playing cards, Roman numerals, and, by allowing the left-most denomination to be negative, two's complement. This ability to place a single perspective on many disparate concepts allows students to achieve a significant amount of mental compression, which as Thurston [5] points out, allows students to recall these concepts quickly and completely.

This mental compression is also related to one the biggest benefits, which is the "stickiness" of student learning. As will be seen in the next section, the class did quite well on final exam, so there is stickiness at the scale of the semester. Students have reported that when they encounter radix representations in other classes they think "It's just ChangeMaker!"

7 Conclusion

Student understanding of *all* radix representations were assessed at the end of the semester via seven questions on the final exam. Recall that the ChangeMaker connection to binary, octal, and hexadecimal was discussed in the course, however, duodecimal, also known as base-12, was not. In spite of this, the final exam assessed student knowledge of binary, octal, hexadecimal,

and duodecimal. In addition, the duodecimal symbols for 10 and 11 devised by William Dwiggins were used, rather than the symbols 'A' and 'B'. Most students were not flustered by having to deal with these symbols which they had never seen before.

The table below reflects the results from two courses containing a total of 42 students and shows very positive student performance. It might be worth noting that a subpopulation of four students missed most of the questions, suggesting that once a student understands the ChangeMaker connection for one radix representation, then they understand it for all. In aggregate, students scored a grand total of 241 points out of 294 for a rate of around 82% overall.

Question	*%Correct*
What is 33_{10} in binary?	95%
What is 33_{10} in octal?	86%
What is 33_{10} in hexadecimal?	81%
What is 27_{10} in duodecimal?	83%
What is 150_{10} in duodecimal?	76%
What is 12_{12} in decimal?	76%
What is 27_{12} in decimal?	76%

8 Acknowledgements

Partial support for this work was provided by the National Science Foundation's Improving Undergraduate STEM Education (IUSE) program under Award No. 1525490. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Boaler. *What's Math Got to Do with It?: How Teachers and Parents Can Transform Mathematics Learning and Inspire Success.* Penguin Books, New York, New York, 2015.

[2] A. diSessa. *Changing Minds: Computers, Learning, and Literacy.* MIT Press, Cambridge, MA, 2000.

[3] S. Papert. Constructionism: A new opportunity for science education. *NSF Grant Proposal*, 1987.

[4] S. Papert. *Situating constructionism.* Ablex Publishing, Norwood, New Jersey, 1991.

[5] W. P. Thurston. Mathematical education. *Notices of the American Mathematical Society*, 37(7):844–850, 1990.

Techniques for Effective Teaching and Learning in the Wake of Transition to Online Classes Due to COVID-19*

Conference Tutorial

Srikantia Subramanya
Visiting Assistant Professor
University of Central Oklahoma
ssubramanya@uco.edu

In the wake of the current COVID-19 pandemic, almost all of the higher educational institutions have transitioned to complete online instruction or some form of hybrid mode where a subset of students of a class attend classes in-person with appropriate physical distancing, and the rest of the students are connected online synchronously. The transition to online mode has resulted in a sudden and drastic reduction in several facilities and features which were available in traditional classes such as a large whiteboard, high level of face-to-face interactions, relatively higher levels of engagement, and longer attention spans, etc. The best of the instructors in traditional classes have suddenly found themselves in uncharted online territory. This is especially challenging in Computer Science courses which contain intricate and complex concepts and have substantial Mathematical content. The online mode of instruction has posed additional challenges in keeping the students motivated, interactive, focused, and engaged during online lectures. Conventional teaching methods are not easily translated to the online context in terms of content delivery and keeping the students engaged. To make up for these, and to keep the effectiveness and teaching and the learning experience at desirable levels, there is a dire need for new set of techniques. The proposed workshop discusses a few techniques for use in online instruction which has been found to be effective in delivering content and in keeping student interactions and engagement at desirable levels. Some of the techniques which will be covered in workshop include:

*Copyright is held by the author/owner.

- Making connections to real–world applications
- Having students come up with questions, given facts
- Having students come up with counter examples
- Posing conventional problems with a slight twist
- Use of relevant history / trivia / quotes / etc.
- Having students come up with precise problem statement, based on some scenarios
- Having students detect (deliberately introduced) errors
- Making connections between different, but related problems
- "Building" algorithms using predefined computing "components"

Although this workshop uses examples related to a particular course in Computer Science, namely Algorithm Design, several of the ideas and concepts can be easily adapted to other Computer Science courses, and also possibly to courses in other disciplines. It must be noted that the effectiveness of the technique discussed here is based on informal feedback and discussions. A detailed formal study would be a worthy future work.

Enrich Student Learning Experience by Building a Cybersecurity Virtual Lab with Open-Source Tools*

Conference Tutorial

Jianjun Zheng
Department of Computer Science
Stephen F. Austin State University, Nacogdoches, TX 75962
jeffrey.zheng@sfasu.edu

A lab session is an important component of a successful and effective cybersecurity curriculum in computer science. It gives students the opportunity to apply the cyber defense knowledge they have learned in the classroom into practice. Moreover, students in a lab session tend to be more interactive and engaging and often have a successful and effective learning experience. Course instructors who teach cybersecurity courses usually rely on textbook authors or publishers to provide some type of virtual lab. While this approach is convenient and time-efficient, as instructors can focus more on preparing the course, it has some challenges when adopting this approach.

First of all, it potentially limits the textbook selections for a course because the course instructors tend to choose a textbook that comes with a lab option, even when there is a better textbook candidate that lacks a lab option. Secondly, the assignments in the provided virtual labs are predefined and instructors do not have the flexibility to modify them if needed. Thirdly, a textbook with a lab option costs more than a regular textbook and could impose a financial burden on students and even deter some students from taking the course.

To address the aforementioned challenges, this workshop demonstrates how to use available open-source tools to create a virtual cybersecurity and networking lab. There is a wide selection of well-known and open-source tools available that have been used in cybersecurity education and training, and it can give course instructors full control over the setup of the lab environment to suit their teaching objectives, without having to be tied with a specific textbook for the sake of a lab option. Since the lab is built with open-source and free tools, students will not be financially burdened and will be more willing to take the course and engage in it when they are given the chance to build the lab themselves.

*Copyright is held by the author/owner.

Play With Trained Hierarchical Reinforcement Learning Agents in Two Common Games*

Conference Tutorial

Chengping Yuan, Mark V. Albert, Daniel McGartland
Jakob Smith, Anthony Solorio
Department of Computer Science & Engineering
University of North Texas
Denton, TX. 76203
chengpingyuan@my.unt.edu, mark.albert@unt.edu

Participants will have an opportunity to learn the basic concepts of reinforcement learning. They can engage with a recently developed reinforcement learning system to play two different games, Connect 4 and Tic-Tac-Toe, with reinforcement learning agents. The games are hosted on a mobile platform (Android, iOS). Participants can explore tactics of how to play individual games as well as the strategy of how to bet or withdraw optimally in matches of multiple games with a variety of opponents. Each game can be set at three different difficulty levels; both strategic and tactic levels learning have their own difficulty levels that can be set separately. Participants can learn that the dual-level learnings are trained with the same algorithm. A quick, broad overview of reinforcement learning will lead to a summary of how this dual-level reinforcement learning model differs from standard game-playing models. The designers of the system are UNT undergraduate and graduate students. The designers will assist participants in setting up and running the games. Participants will be able to observe the flexibility and power of reinforcement learning.

The target audience includes anyone with interest in machine learning. A workshop like this can stimulate early computer scientist students to pursue more challenging aspects of machine learning. In particular, it is a stark reminder of the power of abstraction as the same reinforcement learning model explains two levels of decision making for individuals and adapts flexibly to

*Copyright is held by the author/owner.

a variety of games. For those students who have taken machine learning, often exposure to reinforcement learning is limited to one week, with minimal time to fully demonstrate the capabilities of these systems. The fact that reinforcement learning is a large part of what drives human and animal decision making is often lost by having to cover the technical details of Q-learning or similar methods in the limited time available. This workshop will also provide them an opportunity to ask detailed questions on this more advanced reinforcement learning strategy and allow them to relate to their exposure in prior coursework.

On-ramp to AI: lessons from the introductory AI course "Software Development for AI"*

Conference Tutorial

Ting Xiao[1] and Mark V. Albert[1]
[1]Department of Computer Science & Engineering
University of North Texas
Denton, TX. 76203
ting.xiao@unt.edu, mark.albert@unt.edu

This workshop will discuss the benefits and lessons learned in a project-oriented first course in software development. The intended audience for the workshop is similar to the makeup of the course under discussion - STEM-oriented students interested in a career in Artificial Intelligence. Much like learning a foreign language, experiential learning is a necessary component of learning how to create software, however, students pursuing Artificial Intelligence at the graduate level come from a variety of backgrounds. Students in the course learned tools and techniques focused on three application areas: 1) efficient data and model exploration using Jupyter notebooks 2) flexible use of cloud APIs from AWS, GCP, and Azure, and 3) Mobile/Front-end development.

The University of North Texas had the first cohort of Masters in Artificial Intelligence students in Fall 2020. The inaugural class of approximately 40 students was approximately half professionals in areas of software development, one quarter computer science students, and one quarter STEM students with limited programming experience. However, a project-focused effort led to sharing of expertise through four separate project periods where students are required to mix with new students on new projects in each iteration. This led to 45 different projects with a variety of application domains including 17 in computer vision, 15 in natural language processing , and 7 in audio processing including speech and music. Basics of source code management, data structure, and python were covered with a primer in machine learning for navigating

*Copyright is held by the author/owner.

the terminology of AI APIs used. However, the emphasis was on processes including web and UI design principles, job ad keyword overviews, and open source management of both source code and teams so that base principles were efficiently and effectively put into practice. Participants in the workshop will hear the variety of project experiences of students building their skills in software development with a focus on integrating AI frameworks and APIs. Participants outcomes include a greater awareness of tools and techniques in developing software and experiential approaches to learning how to become actively engaged in creating AI-driven software.

Programming With the Cloud*

Conference Tutorial

Laurie White
CS Professor Emeritus, Mercer University
Developer Relations Engineer, Google Cloud
Former Program Chair, CCSC:SE
lauriewhite@google.com

While there's a lot to learn about cloud computing, the cloud can also be used in classes as fundamental as programming courses with little change to the material being taught. The cloud can provide a uniform programming environment for students regardless of the computers they use to access it remotely. It can provide computing resources beyond what some students may have on their own computers. And there are even some cloud services that can be used to make even the simplest programming assignments more interesting.

*Copyright is held by the author/owner.