The Journal of Computing Sciences in Colleges

Volume 27, Number 4 **April 2012**

The Journal of Computing Sciences in Colleges

Papers of the Twenty Third Annual CCSC South Central Conference

April 20-21, 2012
West Texas A&M University
Canyon, Texas

Papers of the Fifth Annual CCSC Southwestern Conference

March 23-24, 2012
University of the Pacific
Stockton, California

Volume 27, Number 4 **April 2012**

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges. Printed in the USA. POSTMASTER: Send address changes to Paul Wiedemeier, CCSC Membership Chair, Assistant Professor of Computer Science, The University of Louisiana at Monroe, Computer Science and CIS Department, 700 University Avenue, Administration Building, Room 2-37, Monroe, LA 71209.

TABLE OF CONTENTS

THE CONSORTIUM FOR COMPUTING SCIENCES IN COLLEGES BOARD OF DIRECTORS

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

Bob Neufeld, President (2012), P. O. Box 421, North Newton, KS 67117, 316-283-1187 (H), neufeld@mcpherson.edu.

Laura J. Baker, Vice-President (2012), Computer Sciences Department, St. Edward's University, Box 910, 3001 S. Congress Ave, Austin, TX 78704, 512-448-8675, laurab@stedwards.edu.

Paul Wiedemeier, Membership Chair (2013),The University of Louisiana at Monroe, Computer Science and CIS Department, 700 University Avenue, Monroe, LA 71209, 318-342-1856 (O), 318-342-1101 (fax), ccscpaul@gmail.com.

Bill Myers, Treasurer (2014), Consortium for Computing Sciences in Colleges 100 Belmont-Mount Holly Road Belmont Abbey College Belmont, NC 28012-1802, (704) 461-6823, FAX: (704) 461-5051, myers@crusader.bac.edu.

John Meinke, Publications Chair (2012), US Post: CMR 420, Box 3368, APO AE 09063; (H) Werderstraße 8, D-68723 Oftersheim, Germany, 011-49-6202-5 77 79 16 (H), meinkej@acm.org.

Colleen Lewis, Southwestern Representative (2014), University of California at Berkeley, 329 Soda Hall, Berkeley, CA 94720,510-388-7215, colleenl@berkeley.edu.

Elizabeth S. Adams, Eastern Representative (2014), James Madison University - Mail Stop 4103, CISAT - Department of Computer Science, Harrisonburg, VA 22807, 540-568-2745 (fax), adamses@jmu.edu.

Jeff Lehman, Midwestern Representative (2014), Mathematics and Computer Science Department, Huntington University, 2303 College Avenue, Huntington, IN 46750, 260-359-4209, jlehman@huntington.edu.

Scott Sigman, Central Plains Representative (2014), Drury University, 900 E. Benton Ave., Springfield, MO 65802, 417-873-6831, ssigman@drury.edu.

Kevin Treu, Southeastern Representative (2012), Furman University, Dept of Computer Science, Greenville, SC 29613, 864-294-3220 (O), kevin.treu@furman.edu.

Timothy J. McGuire, South Central Representative (2012), Sam Houston State University, Department of Computer Science, Huntsville, Texas 77341-2090, 936-294-1571, mcguire@shsu.edu.

Brent Wilson, Northwestern Representative (2012), George Fox University, 414 N. Meridian St, Newberg, OR 97132, 503-554-2722 (O), 503-554-3884 (fax), bwilson@georgefox.edu.

Pat Ormond, Rocky Mountain Representative (2013), Utah Valley University, 800 West University Parkway, Orem, UT 84058, 801-863-8328 (O), 801-225-9454 (cell), ormondpa@uvu.edu.

Lawrence D'Antonio, Northeastern Representative (2013), Ramapo College of New Jersey, Computer Science Dept., Mahwah, NJ 07430, 201-684-7714, ldant@ramapo.edu.

Linda Sherrell, MidSouth Representative (2013), University of Memphis, 209 Dunn Hall, Memphis, TN 38152, 901- 678-5465 (O), linda.sherrell@memphis.edu.

Serving the Board: The following CCSC members are serving in positions as indicated that support the Board:

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

George Benjamin, Associate Editor, Muhlenberg College, Mathematical Sciences Dept., Allentown, PA 18104, 484-664-3357 (O), 610-433-8899 (H), benjamin@muhlenberg.edu.

Susan Dean, Associate Editor, US Post: CMR 420, Box 3369, APO AE 09063; Werderstraße 8, D-68723 Oftersheim, Germany. 011-49-6202-5 77 82 14 (H), sdean@faculty.ed.umuc.edu.

Robert Bryant, Comptroller, MSC 2615, Gonzaga University, Spokane, WA 99258, 509-313-3906, bryant@gonzaga.edu.

Mark Goadrich, National Partners Chair, Centenary College of Louisiana, 2901 Centenary Boulevard, Shreveport, LA 71104, 318-869-5194, mgoadrich@centenary.edu.

Brent Wilson, Database Administrator, George Fox University, 414 N. Meridian St, Newberg, OR 97132, 503-554-2722 (O), 503-554-3884 (fax), bwilson@georgefox.edu.

Myles McNally, Webmaster, Alma College, 614 W. Superior St., Alma, MI 48801, 989-463-7163 (O), 989-463-7079 (fax), mcnally@alma.edu.

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.

Turings Craft

National Science Foundation

Wiley

FOREWORD

My sincere thanks and commendations to the program committees for the two conferences represented in this issue of the *Journal*. Both committees have worked well together with lots of communication putting together an excellent program as well as facilitating the papers of those conferences appearing in this issue in a very efficient fashion. We work through the regional editors and when I have a question and I get an almost immediate response it facilitates the final editing process being accomplished very efficiently. My sincere thanks to Laura Baker (South Central) as well as Peter Gabrovsky and Mike Doherty (Southwestern) for being so wonderful to work with. This is particularly true since we were working over a span of nine time zones – one was going to bed while another was beginning the day. My sincere thanks to Laura, Peter, and Mike for being so on top of things. I also need to give appropriate credit to Susan Dean who is there when we have a "surplus" of work. We had two sets of manuscripts arrive within 24 hours for two of the Spring conferences, neither of which was for this issue but also sharing tight printing deadlines. Susan took over on one conference while I worked on the other set, and we were feeding each other back and forth with manuscripts from the other Spring conferences. I also very much rely on Susan for proofreading. It always works best when there are fresh eyes on the final copy, so rely on her to proof what I have reformatted while I proof what she has reformatted. It's a real team concept. I also must thank Bob Neufeld who helps with the proofing. No matter how many times we proof an issue there are small items that slip through, and we refer to Bob as "Eagle Eye." He manages to find lots of small items that slip past both of our eyes!

We begin our coverage of the Spring set of conferences with this issue of the *Journal*. The first Spring conference is in late March and the five conferences are then all occurring over a five week period, the last being the last weekend in April. They are spread geographically from the West Coast to the Northeast. There is an excellent mix of professional papers and tutorials/workshops/panel discussions in each. I commend all of them, and encourage you to plan appropriately so that when there are multiple conferences that you are able to participate in that cross-fertilization between regions. While CCSC sponsors regional conferences the involvement across regional boundaries

really adds to the overall professional experience.

I know that you will benefit greatly from the contents of this issue of the *Journal* as well as the two that will follow, and those of you who attend the conferences will benefit that much more.

The papers in the *Journal* are all reviewed and the reviewing process as well as the acceptance rate are outlined in the welcome statements to each of the individual conferences. The reviewing process is double blind which means that the reviewer doesn't know whose paper he/she is reviewing plus the author doesn't know who has reviewed the manuscript. This helps to maintain the quality of the submissions.

Our list of CCSC National Partners is a listing of major contributors to the conferences. Without their help it would be necessary to raise conference registration rates, and CCSC has long been aware that budgets are tight. Maintaining low registration costs assists those with low travel budgets to continue their involvement in professional development in the computing sciences. Please be certain to thank our National Partners for their involvement and support of CCSC.

We also need to recognize the support of Upsilon Pi Epsilon, the National Honor Society for Computer Science. For a good number of years now they have been financially supporting CCSC conferences through support of student participation. It is through their assistance that we can have student oriented activities at the conferences. Our sincere thanks to UPE.

As this issue of the *Journal* goes to press we also say good-bye to our former printer, Montrose Publishing. They have been part of the CCSC "family" for a quarter century. Founded in 1816, they recently closed their doors. We thank them for their service to CCSC over many good years, and wish them well. This also marks a new beginning as we welcome Courier Printing. "Good-bye" to some old friends and "Welcome" to new friends. The expression in German is mit jede Ende kommt ein neuer Anfang.

Again, it is a pleasure to present this issue in this year's edition of the *Journal of Computing Sciences in Colleges*. I trust that you will find the contents as interesting and worthwhile as I have.

John Meinke, UMUC Europe
CCSC Publications Chair

Papers of the Twenty Third Annual CCSC South Central Conference

April 20-21, 2012
West Texas A&M University
Canyon, Texas

WELCOME TO THE TWENTY THIRD ANNUAL (2012) CCSC SOUTH CENTRAL CONFERENCE

Welcome to the 23rd annual conference of the South Central Region of the Consortium for Computing Sciences in Colleges hosted by West Texas A&M University in Canyon, Texas. The South Central Steering Committee is pleased to provide a scenic venue in the canyons of the panhandle of Texas. The Steering committee would like to extend our sincere thanks to all of the authors, presenters, speakers, attendees, and students for participating in the conference. The faculty, staff, and students at West Texas A&M have provided a terrific venue for the region and special thanks to our conference chair, Rajan Alex, who has worked tirelessly on behalf of the conference and the Consortium.

This year ten professional papers and six workshops were accepted for publication and presentation. We use a double-blind paper review process and each paper is reviewed by three independent reviewers. There were 15 papers submitted and 10 were accepted for publication for a 66% acceptance rate. Thirty-four professionals provided reviews of papers for this year's conference. We really appreciate the time and effort put into our reviewing process by all of the reviewers from across the United States and Portugal.

On behalf of the Steering Committee we welcome our members to encourage colleagues and students to participate in the conference in the future and to consider serving on the conference Steering Committee.

Laura J. Baker
Papers/Program Chair, South Central
St. Edward's University

Rajan Alex
2012 South Central Conference Chair
West Texas A&M University

2012 SOUTH CENTRAL STEERING COMMITTEE AND CONFERENCE COMMITTEE

Ken Hartness, 2011 Conference Chair.. . . . . Sam Houston State University, Huntsville, TX
Rajan Alex, 2012 Conference Chair. . . . . West Texas A&M University, Canyon, TX
Leslie Fife, 2013 Conference Chair. . . . . Louisiana State University - Shreveport, Shreveport, LA
Members with term ending in 2012:
Kay Kussman, 2012 Moderators Chair. . . . . McNeese State University, Lake Charles, LA
Leslie Fife, 2012 Conference Registrar. . . . . Louisiana State University - Shreveport, Shreveport, LA
Phyllis Tedford, South Central Region Webmaster.. . . . . Texas A&M University - Corpus Christi, Corpus Christi, TX
Members with term ending in 2013:

Michael Kart, 2012 Treasurer. . . . . St. Edward's University, Austin, TX
Vipin Menon, Publicity Chair. . . . . McNeese State University, Lake Charles, LA
Michael Scherger, 2012 Student Papers Chair.. . . . . Texas A&M University - Corpus Christi, Corpus Christi, TX
Members with term ending in 2014;
Anne-Marie Eubanks, Panels and Tutorials Chair.. Stephen F. Austin State University
Shyam (Sam) Karrah, 2012 Student Project Posters Chair. UT-Dallas, Richardson, TX
Rakesh M. Verma. . . . . University of Houston, Houston, TX
Other Members;
Laura J. Baker, 2012 Papers/Program Chair. . . . . St. Edward's University, Austin, TX
Timothy J. McGuire, CCSC Board Member (SC Rep). . . . . Sam Houston State University, Huntsville, TX

REVIEWERS — 2012 CCSC SOUTH CENTRAL CONFERENCE

Krishna Agarwal. . . Louisiana State University at Shreveport, Shreveport, Louisiana
Ghassan Alkadi. . . . . Southeastern Louisiana University, Hammond, Louisiana
Pavel Azalov. . . . . Penn State - Hazleton, Hazleton, Pennsylvania
Barbara Anthony. . . . . Southwestern University, Georgetown, Texas
Brent Baas. . . . . LeTourneau University, Longview, Texas
John Camden. . . . . St. Edward's University, Austin, Texas
M. Emre Celebi. . . . Louisiana State University at Shreveport, Shreveport, Louisiana
Ming-Hsing Chiu. . . . . Southeastern Louisiana University, New Orleans, Louisiana
Ernie Giangrande Jr.. . . . . Wingate University, Wingate, North Carolina
Rohitha Goonatilake. . . . . Texas A & M International University, Laredo, Texas
David Gurney.. . . . . Southeastern Louisiana University, Hammond, Louisiana
Ranette Halverson. . . . . Midwestern State University, Wichita Falls, Texas
Tina Johnson. . . . . Midwestern State University, Wichita Falls, Texas
Michael Kart. . . . . St. Edward's University, Austin, Texas
Srinivasarao Krishnaprasad. . . . Jacksonville State University, Jacksonville, Alabama
Kay Kussmann. . . . . McNeese State University, Lake Charles, Louisiana
James McGuffee. . . . . St. Edward's University, Austin, Texas
Lewis Myers. . . . . St. Edward's University, Austin, Texas
Vipin Menon. . . . . McNeese State University, Lake Charles, Louisiana
José Carlos Metrôlho. . . . . Instituto Politécnico de Castelo Branco, Portugal
Jaime Niño. . . . . University of New Orleans, New Orleans, Louisiana
Myung Ah (Grace) Park.. . . . . University of Central Oklahoma, Edmond, Texas
Nelson Passos.. . . . . Midwestern State University, Wichita Falls, Texas
Lakshmi Prayaga. . . . . Pensacola State College, Pensacola, Florida
Yong Shi. . . . . Kennesaw State University, Kennesaw, Georgia
Robert Strader. . . . . Stephen F. Austin State University, Nacogdoches, Texas
Donna Teel.. . . . . University o f Mary Hardin - Baylor, Belton, Texas
Rakesh Verma. . . . . University of Houston, Houston, Texas
Ke Yang. . . . . Louisiana State University, Baton Rouge, Louisiana
Kwok-Bun Yue. . . . . University of Houston - Clear Lake, Houston, Texas

KEYNOTE ADDRESS

Friday, April 20, 2012

AN INDUSTRY VIEW OF COMPUTER SCIENCE/COMPUTER ENGINEERING/SOFTWARE ENGINEERING EDUCATION AND ACCREDITATION [*]

Dennis Frailey, Ph.D.
ACM Distinguished Speaker

Today's competitive companies rely on computing science, computer engineering and software engineering programs to provide highly qualified staff for computer and software development assignments. Accreditation is one way to identify the programs that meet minimum criteria for the profession. This talk provides an overview and critique of the accreditation processes and of computing science and computer and software engineering programs in general from the viewpoint of industry. Also addressed are the qualifications that industry looks for in computing science / computer engineering / software engineering graduates, and some of the likely career options.

INTRODUCTION TO PROGRAMMING WITH THE FINCH ROBOT*

PRE-CONFERENCE WORKSHOP

Anne-Marie Eubanks, Robert G. Strader
Computer Science Department, Stephen F. Austin State University
Box 13063, Nacogdoches, TX, (936) 468-2508
eubanksanne@sfasu.edu, rstrader@sfasu.edu

Robots are becoming a popular learning tool in computer science classes today. They offer active learning for the student, they engage the student's problem solving skills, and are easily applied to collaborative projects. They also allow the teachers to observe how the student is learning the objective. This workshop uses the Finch from Carnegie Mellon's CREATE lab and Jython to create introductory programs for CS0 students. The IDE used is JES (Jython Environment for Students) version created for the Finch. The Finch is an affordable robot and this JES version has the compiler, IDE, and Finch libraries in one convenient download. The participants will be provided with a Finch and two introduction programs.

DESIGNING COURSES FOR HYBRID INSTRUCTION: PRINCIPLES AND PRACTICE *

Rob Wood, Esmail Bonakdarian, Todd Whittaker
Franklin University
201 S. Grant Avenue
Columbus, OH, U.S.A.
{woodr | esmail | whittakt}@franklin.edu

ABSTRACT

Instructional design is for education what software engineering is for computer science; both are the discipline-specific application of processes that result in higher quality outcomes. To that end, we suggest specific instructional design elements for institutions interested in implementing a hybrid format. We use the term hybrid to denote the mixing of online and face-to-face students in the same class, in a way that combines the most engaging aspects of traditional face-to-face delivery with online flexibility. Although they share many common design principles, each of the major instructional formats has unique features, and we integrated those to develop the hybrid model described in this report, which we successfully ran for several terms. The results in this paper yield relevant advice based on theory and practice.

1 INTRODUCTION AND MOTIVATION

Universities and colleges expanding their offerings to online are faced with the challenges of negative public perception of online instructional quality [7] and faculty resistance to teaching online [8]. A hybrid format can address these challenges by providing an experience that is not radically different from a face-to-face (F2F) class yet offers many of the benefits of online instruction. We find that students have judged this format as better than online and at least as good as F2F (Section 3.4.3).

We use the term hybrid to denote the mixing of online and F2F students simultaneously in the same class. This is unlike a "blended" format, which may interleave F2F and online instruction by week, or simply try to augment a traditional F2F class with

---

some online elements, such as tutorials and references. Hybrid, as used here, means that elements from the F2F and online formats are combined such that both populations share the same instructional experience.

The hybrid format provides online students the opportunity to become more engaged with the material, their fellow students, and instructors by allowing them to interact in a manner similar to F2F classes. At the same time, this format offers the convenience of online classes without forcing radical changes on instructors who are used to teaching F2F classes. The hybrid format can also benefit institutions economically. For example, when offering both online and F2F sections of the same upper division course, low enrollment in both sections may occur. In a hybrid format, two low enrollment classes could be combined into one, meeting both student populations' needs and requiring only one instructor.

Section 2 of this paper refers to the use of instructional design methods in higher education and describes the theoretical framework used for our hybrid approach. Section 3 explains the application of these principles in more detail along with specific examples and observed results. The paper concludes in Section 4.

2 THEORETICAL FRAMEWORK

Instructional design (ID) is a systematic approach to creating instruction that facilitates student achievement of learning outcomes and is based on a variety of disciplines including education, psychology, cognitive science, and organizational behavior.

ID uses the same principles for F2F, online, and hybrid courses with the goal of giving all students equal opportunities to achieve the learning outcomes. In practice, this means that learning activities may differ across formats, but the ability of students to achieve the learning outcomes is equivalent.

We employ a "principle-to-practice" approach as the foundation of our hybrid design, applying principles of relevant learning theories systematically to design courses that are student-centered and provide authentic assessment activities in practice. Specifically, we adapted David Merrill's [6] "First Principles of Instruction" to design hybrid courses. Merrill's first principles state that learning is promoted when students solve problems in a real-world context, activate existing knowledge as a foundation for new knowledge, observe the demonstration of new knowledge, and apply and integrate it into their current framework.

3 APPLICATION OF ID THEORY

Based on the principles of the aforementioned framework, we selected pedagogical structures for the hybrid class that fit within our course and technology infrastructure. These structures were repeated every week to lend consistency and predictability for both F2F and online students, as identified below:

- *Activation:* ungraded, pre-class exercises and readings.
- *Demonstration:* synchronous web-meetings for lecture, recitation, and live coding exercises.

- *Application:* ungraded, post-class exercises; graded homework and lab assignments.
- *Integration:* weekly reflective notes and final experiences paper.
- *Real world context:* problems and assignments tailored to relate to working, adult students.

We taught four terms of hybrid classes using these constructs. Students were surveyed to determine the efficacy of each. These data, along with more complete descriptions of the needed technology and student and instructor perceptions, are reported in [9].

To judge the efficacy of the design, we employed the e3 learning framework [5], which focuses on the effectiveness, efficiency, and engagement of the instruction. Paraphrasing Doering and Veletsianos[4], these three attributes were defined for students as:

- *Effective:* helped me to learn
- *Engaging:* held my interest
- *Efficient:* used my time well

The approach was piloted over four semesters in two different undergraduate courses using different instructors. The courses were Object-Oriented Data Structures & Algorithms II and Application Server Programming, with most data gathered from the first course. We successfully used this approach in several other courses (technical communication, web development, etc.) at the junior and senior levels. The applicability of this format to large, lecture-based introductory courses remains to be seen as we have no such courses at our institution.

We surveyed 29 students, 21 of whom responded (the relatively small number of respondents is due to our initial pilots of the hybrid format as an alternative to canceling low-enrollment F2F sections). Roughly equal numbers of F2F and online students are represented in the data. Students evaluated each structure using a 5-point Likert scale, where 5 represented a highly effective, efficient, or engaging structure, and 1 represented an utterly futile, wasteful, or boring structure. In addition, we read through students' final reflection papers and coded their responses using the same categories as the survey.

3.1 Activation of Prior Knowledge

3.1.1 Principle

The activation phase guides our design of activities that encourage students to demonstrate what they already know as a foundation for the new knowledge they are beginning to acquire. We used pre-class activities to accomplish activation of prior knowledge (activities range from out-of-class reading assignments to self-assessments to pre-tests that reveal gaps in students' understanding of fundamental concepts). In-class pre-learning activities can be as simple as the instructor providing students with a series of cues and questions to activate prior learning. These should focus on important concepts, particularly those with which students are known to have difficulty.

3.1.2 Practice

The activation phase used pre-class activities that were to be attempted by individual students after the required weekly readings, but before the synchronous sessions. A mix

of short answer questions, guided examples, problem-based learning, and storytelling was used. Each exercise mapped to a specific learning outcome for a given week, and exercises were based as much as possible on real-world situations. These problems were ungraded.

3.1.3 Results

Students found activation exercises to be the least effective, efficient, and engaging structures. Many students initially attempted the exercises but under time pressure as the term advanced, they gave up trying. Those who did continue found themselves better prepared for subsequent phases. Overall, the pre-class activities were perceived as relatively effective but somewhat boring. Table 1 groups the five Likert-scale responses into positive, neutral, and negative categories.

	Positive	**Neutral**	**Negative**
Effective	43%	52%	5%
Efficient	43%	38%	19%
Engaging	29%	48%	24%

Table 1: Pre-class exercise ratings

3.2 Demonstration of New Knowledge

3.2.1 Principle

This phase requires that we design instruction that demonstrates what students will learn. In short, students do not want simply to be told - they want to be shown as well. One useful method is to intersperse lectures with guided activities in which the instructor demonstrates a problem-solving approach and then introduces students to a similar situation that requires them to apply what they have just learned. A key to designing for the demonstration phase is engagement and interaction for the online students. Demonstrations, whether live or media-based, must be clear, easy to follow, and provide students with the information they need to observe and practice the newly learned material.

3.2.2 Practice

The demonstration phase is at the heart of a hybrid course and is represented in our format by synchronous recitations conducted both F2F and online simultaneously via web-meeting software. A typical session consisted of a repeating sequence: identification of the learning outcome, a short content presentation, the solution to the associated pre-class learning activity, the solution to a more in-depth problem, and a brief discussion. Each sequence took approximately 20 minutes. Some of these sequences were slide-based problems, but several involved a start-to-finish coding demonstration with related test cases. In addition, the demonstrations gave students opportunities to discuss solutions and participate in the problem solving process - something not easily

implemented in asynchronous online courses. For information about the specific technologies used to conduct these sessions, see [2] and [3].

Synchronous participation from the online students was not required, but encouraged by assigning points to one of two activities: students could either attend the live session, or review the recordings and write a short summary and reflection. Most students opted to attend the live session.

3.2.3 Results

Since it is the major distinction between F2F, asynchronous online, and hybrid classes, we asked about several components of the demonstration phase: the synchronous sessions (Table 2), the live coding demonstrations (Table 3), and the recordings of the previous two structures (Table 4).

It is easy to see that an overwhelming majority of students found the synchronous sessions to be highly useful to their learning (Table 2). This confirms that online students would prefer to have an experience more like F2F instruction but cannot do so for various reasons [1]. Hybrid instruction may hit the "sweet spot" for these students.

	Positive	**Neutral**	**Negative**
Effective	90%	5%	5%
Efficient	86%	10%	5%
Engaging	76%	19%	5%

Table 2: Synchronous session ratings

	Positive	**Neutral**	**Negative**
Effective	67%	24%	10%
Efficient	38%	33%	29%
Engaging	33%	38%	29%

Table 3: Live coding demonstration ratings

Live coding examples were also an integral part of the demonstration phase since they showed the problem solving process as much as they did the solution. Interestingly, students indicated that this helped them to learn but was not a particularly interesting or wise use of their time (Table 3). Student comments indicated that they found it discouraging to see something solved in 20 minutes that took them five or six hours to complete.

Student ratings of the recorded sessions were similar to those of the live synchronous sessions (Table 4). In their reflection papers, many students (including those who attended F2F) reported reviewing those recordings as part of their exam preparation activities. Recorded sessions were also important for those online students who could not attend live for various reasons.

	Positive	Neutral	Negative
Effective	71%	14%	14%
Efficient	76%	10%	14%
Engaging	76%	14%	10%

Table 4: Recorded session ratings

Finally, students found the additional instructor interaction afforded by the demonstration phase in the hybrid format to be highly beneficial but did not value the interactions with other students as much. Several online students commented that participating in this phase with the F2F students increased their sense of community, but they did not associate that directly with increased learning.

3.3 Application of New Knowledge

3.3.1 Principle

It is critically important that the acquisition of new knowledge is immediately followed by its application. We design the application phase with the principle that students must be able to use newly acquired knowledge to solve problems. Our application-oriented post-class activities, homework assignments, and programming exercises reinforce what students have learned and give them authentic opportunities to practice new knowledge in a relatively low-stakes manner. It is important to note that we do not rule out written assessments when they have direct relevance to the learning, or when they can serve as diagnostic tools to measure student learning from a formative perspective.

Feedback from the instructor to the students during the application phase must be planned, diagnostic, and consistent; therefore we ensure that appropriate feedback mechanisms are in place.

3.3.2 Practice

The application phase consisted of post-class active learning exercises and homework/programming problems (lab assignments were more integrative and deferred to the last phase). Post-class activities built on the knowledge in the demonstration phase, and homework built on the post-class activities.

These activities were at a level comparable to the more in-depth problems presented in the synchronous sessions and similar to the pre-class activities in that they did not contribute to the students' marks in the class. Most importantly, these activities occurred immediately after the synchronous session; F2F students worked on them for the balance of the class time (up to two hours) while online students had the opportunity to remain in the web-meeting and do likewise (although this was not required). Monitored post-class activities directly combat the "big blank screen" syndrome that plagues programming students who, while claiming to understand everything in recitation, do not know where to start when solving a problem on their own.

3.3.3 Results

The majority of students found the post-class active learning exercises to be effective, efficient, and engaging (Table 5). This result is interesting because the post-class exercises were very similar to those in the activation phase, yet they were rated higher.

	Positive	Neutral	Negative
Effective	57%	24%	19%
Efficient	57%	19%	24%
Engaging	52%	29%	19%

Table 5: Post-class exercise ratings

One possible explanation for this seeming discrepancy is that students were not actually attempting the pre-class active learning exercises and instead were using the demonstration phase for both activation and demonstration. Without assigning grades to the activation phase exercises (which would raise the stakes for students and increase the grading burden for instructors), there is little that can be done. Another possible explanation is the availability of instructor coaching for the post-class activities. Since these activities were monitored by the instructor, many students worked on them and were afforded the opportunity to ask questions.

Homework problems concluded the application phase, but these do not differ significantly from what is typical in an undergraduate computer science class.

3.4 Integration of New Knowledge

3.4.1 Principle

Merrill [5] stated, "A learning cycle is completed when learners have an opportunity to integrate new knowledge and skill into their everyday activities." For adult students (the primary student population at our institution), it often means that they attempt integration of their knowledge in their workplaces. Our experience has shown that significant integration frequently occurs long after students have departed their classrooms and are practicing their knowledge and skills in their professions.

3.4.2 Practice

The integration phase concluded each week's activities. We used a combination of summative assessment via lab assignments and formative assessment via weekly ungraded reflection and graded end-of-term reflection papers. Lab assignments were not substantively different from those found in many computer science courses; however, care was taken to ensure that the problems were more challenging, covered many (if not all) of the learning outcomes, and incorporated the specific skills taught as well as real world applications.

These weekly reflections culminated in an end-of-term reflection paper: a progressive, personal account of the students' learning for an entire term. To help students

focus on describing their actual learning, we provide reflection trigger questions that serve to guide their thinking (e.g., "What skills and concepts have I used to successfully complete the learning activities?" and "Has my earlier understanding changed in light of the new material?"). Students did not submit weekly reflection notes for grading; however, those who did write weekly reflections saw more connections among the material.

3.4.3 Results

The reflection papers contain valuable anecdotal data on what works and what does not work in a course (or in this case, a delivery format). Comments in the papers indicated that the design of the hybrid class worked well and students successfully integrated new knowledge from the course into their view of the field. Students were surprised at how much they had learned and expressed an overwhelming preference for hybrid over standard online (Table 6) and rated it at least as good as F2F (Table 7).

	Positive	**Neutral**	**Negative**
Effective	71%	14%	14%
Efficient	71%	14%	14%
Engaging	76%	14%	10%

Table 6: Preference for hybrid over online

	Positive	**Neutral**	**Negative**
Effective	48%	43%	10%
Efficient	62%	33%	5%
Engaging	43%	48%	10%

Table 7: Preference for hybrid over face-to-face

4 CONCLUSIONS

Based on our experience over four terms, we believe that our hybrid approach offers institutions a viable, low risk alternative to strictly online instruction. Hybrid combines the most engaging aspects of face-to-face delivery with online flexibility. We were gratified to see student comments like the following (taken from the end-of-term course evaluations), reflecting our goals for the hybrid format:

"...I told myself I wasn't going to take another online course again if I could help it... I have to admit that had I known that this class would have been run in the structure it was, I would have had no issues with having to take this class online. The hybrid format is a good format and should be the standard [at our institution]."

We believe that our practices and results are relevant for and transferable to other institutions. We hope that others will explore this format and share their experiences with the larger teaching community.

Finally, at least in part based on this work, our institution has approved this hybrid approach as a standard format in addition to F2F and online.

5 ACKNOWLEDGMENTS

The authors would like to thank Jillian Evans for her most expert editorial help with this paper.

6 REFERENCES

[1] Students prefer hybrids to fully online courses. *Recruitment and Retention in Higher Education,* 19(8):7-8, August 2005.

[2] E. Bonakdarian, T.Whittaker, and D. Bell. Merging worlds: When virtual meets physical - an experiment with hybrid learning. In *The Journal of Computing Sciences in Colleges,* vol 25.1. Consortium for Computer Sciences in Colleges, October 2009.

[3] E. Bonakdarian, T.Whittaker, and Y. Yang. Mixing it up - more experiments in hybrid learning. In *The Journal of Computing Sciences in Colleges,* vol 25.4. Consortium for Computer Sciences in Colleges, April 2010.

[4] A. Doering and G. Veletsianos. What lies beyond effectiveness and efficiency? Adventure learning design. *Internet and Higher Education,* 11(3-4):137-144, 2008.

[5] D. Merrill. *The e-Learning Handbook: A Comprehensive Guide to Online Learning,* Converting e3-learning to e3-learning: An alternative instructional design method, pages 359-400. Pfeiffer/Jossey-Bass, California, 2008.

[6] D. Merrill. *Trends and issues in instructional design and technology,* First principles of instruction: A synthesis., pages 62-17. Pearson Education, 2007.

[7] K. Parker, A. Lenhart, and K. Moore. The Digital Revolution and Higher Education. `http://pewsocialtrends.org/2011/08/28/the-digital-revolution-and-higher-education/1/`, August 28, 2011, Retrieved August 30, 2011.

[8] G. Schell. Universities marginalize online courses. *Communications of the ACM*, 47(7):53-56, 2004.

[9] T. Whittaker and E. Bonakdarian. Face-to-face experiences for online students: effective, efficient, and engaging hybrid classes. In *The Journal of Computing Sciences in Colleges,* vol 26.4. Consortium for Computer Sciences in Colleges, April 2011.

TECHNOLOGY-SUPPORTED COLLABORATIVE SERVICE LEARNING IN UNDERGRADUATE COMPUTER SCIENCE COURSES*

Gwendolyn Walton
Department of Mathematics & Computer Science
Florida Southern College, Lakeland, FL 33801
863-680-6283
gwalton@flsouthern.edu

ABSTRACT

This paper describes a project to investigate the impact of tablet laptops, wireless communication, and collaboration software to enhance student collaborative work on service learning projects in two undergraduate computer science courses (Database Analysis & Design, and Software Engineering) at a small private college. Even though very little instruction was provided to the students on how to use the collaborative technology, the students and faculty reported that the technology had a positive impact on student project participation, a positive impact on students' on-task communication and engagement with the course content, and a positive impact on quality of student work products.

1.0 INTRODUCTION

The ideas of experiential, problem-based learning accompanied by reflection are certainly not new. Dewey (1938) was an early proponent of these ideas.[4] Kolb (1984) extended the early ideas to develop a model of experiential learning.[6] In recent years there have been many calls for improved delivery of undergraduate learning, including recommendations that undergraduate students be provided an opportunity to address complex, real-world problems before they graduate. For example, [3] stated that "good practice in undergraduate education" does the following: "encourages contact between students and faculty, develops reciprocity and cooperation among students, encourages

---

active learning, emphasizes time on task, communicates high expectations, and respects diverse talents and ways of learning." The Association of American Colleges and Universities provided a list of "essential learning outcomes" for the 21st century that includes "Intellectual and Practical Skills, including inquiry and analysis, critical and creative thinking, written communication, … , and teamwork and problem solving", and "Personal and Social Responsibility, including civic knowledge and engagement local and global, …" [1]

The use of collaborative service learning projects that involve analysis and design for a real-world project within the context of a course's student learning objectives is consistent with the calls for improved delivery of undergraduate learning. Many authors (e.g., [2, 3, 5, 7-10]) have reported positive results when incorporating service learning projects in undergraduate computer science database and/or software engineering courses. [7, 8, 9] reported that, when compared to similar courses that included instructor-created projects, service learning courses were characterized by increased student involvement and interest, increased engagement and participation, and improved student understanding of course principles and concepts.

[5, 7, 8] reported some of the challenges of service learning: increased responsibility and additional time required of the instructor; the need for the instructor to carefully control the project scope; and the difficulty in working with an external customer to define a project with close correspondence to the course's student learning objectives. The author's experiences with using service learning projects in undergraduate computer science courses have been consistent with the results reported in the referenced papers. Service learning projects can provide excellent real-world experience for computer science students. Students work together to plan and manage the project; investigate their customer's problem domain; integrate, apply, and reflect on concepts from previous courses; and reflect on the potential impacts of their technology and design decisions. However, service learning projects require a significant amount of planning, coordination, and collaborative work on the part of the both the instructor and the students. Effective collaborative, student-centered learning cannot be effectively managed by an instructor who spends the entire class standing at the front of the class. Cell phones, tablets and other computers, wireless access, and collaboration software can be misused by a student who is easily distracted. However, it has been the author's experience that misuse of technology can be minimized if the instructor moves around the room and acts as a mentor who facilitates the collaboration of each small group. If the students are divided into small groups, every student can be given responsibility for, and be held accountable for, planning and managing some portion of the project work and completing some of the project deliverables. To promote group cohesion and collaboration, students often require mentoring on understanding and valuing their colleagues' different perspectives and abilities.

2.0 THE PROJECT

A Technology for Teaching grant awarded to Florida Southern College by Hewlett-Packard Corporation provided an opportunity to investigate some of the potential benefits of enhancing collaboration in service learning projects through the use of tablet laptops, wireless communication, and collaboration software. The overall goal of the

grant project was to use mobile technology to provide a more student-centered, collaborative learning environment in selected computer science, environmental science, and mathematics courses. Specific project objectives for computer science courses were: (a) Improve student participation and quality of student work on collaborative projects; (b) Increase the frequency of students' on-task communication with one another during class, and (c) Increase the percent of time that students are on-task and actively engaged during class. This paper describes the impacts of the study on two undergraduate computer science courses (Database Analysis & Design, and Software Engineering.)

Prior to the technology enhancement supported by the grant, both of these courses were taught in a classroom that contains 25 stationary workstations. Each workstation has one desktop PC with an Ethernet connection. No wireless access is available in the classroom, and workstations cannot be reconfigured to better support collaborative student work. Both the Database course and the Software Engineering course require the students to collaborate on a service-learning project. Prior to the grant, the only in-class tools available to the students in these classes were: pencil and paper, whiteboards, digital cameras, stationary workstations with individual desktop computers, and student cell phones. Students often had difficulty coordinating their work and compiling their individual efforts to create a professional group product, and some students did not participate fully during in-class small group exercises.

After the grant was awarded, a previously unused physics laboratory was reassigned for use as a computer systems classroom and laboratory. The grant provided 21 tablets in a laptop cart, a projector, digital camera, and a graphics printer for this new classroom. A variety of software was installed on the tablets, including digital ink, Microsoft Office Enterprise Suite, Washington University's Classroom Presenter, Java, C++, Eclipse, MySQL Database Server, and web browsers. Additional funding from the grant was used to provide wireless access in this classroom and in surrounding areas. All class sections involved in the grant project were taught in this new computer systems classroom.

Two sections of the Database Analysis & Design course (Spring 2009 and Spring 2010, a total of 31 students) and two sections of the Software Engineering course (Spring 2009 and Spring 2011, a total of 20 students) were the focus of the portion of the grant project described in this paper. All four of these sections were taught by a single faculty member who uses active, engaged learning pedagogy throughout every class. Both the database course and the software engineering course require that the students work as a group to complete a "real-world" collaborative service learning project that includes analysis, design, development, documentation, and testing. Students in each section of the database course work in small groups over a period of several weeks to produce the following products: report of database design issues and constraints, data dictionary, logical database design, relational database design, sample queries, and a relational database implemented in MySQL or Microsoft Access and populated with artifacts obtained from the intended user. Students in each section of the software engineering course work in small groups on a semester-long real-world project to produce the following products: concept of operations document, software requirements specification, a variety of exploratory prototypes to investigate technology and/or concepts for which the students had no previous experience, high-level architecture and design document, detailed design for the first increment of the software, implementation of the first increment of the software, and complete project documentation.

The students were already skilled in the use of a computer to create and edit documents, and they were enthusiastic about using technology to facilitate collaboration. The use of the pen and digital ink with the tablet PCs was so intuitive that no formal instruction was needed. To introduce the collaboration software, the instructor gave a very short demonstration on how to set up a small group session to electronically share digital notebooks and other electronic documents, and how to change a document while the document was being edited by another student. After a few minutes of free exploration, the students quickly developed their own processes for using the technology to enhance communication to support collaborative work in a small group to develop and edit technical diagrams and other specifications, share work products with other students, and integrate, review and modify work products as needed to efficiently develop project artifacts.

3.0 DATA ANALYSIS AND RESULTS

For each of the course sections impacted by the HP grant, data were analyzed in the following four areas: (a) student perception of their participation on group work and the participation of their team members, (b) student on-task communication and engagement, (c) faculty perception of the quality of each group's final products, and (d) student satisfaction with the instructor and course. Data used in the analysis were obtained via instructor observations during in-class work, instructor assessment of final projects, student's peer reviews, and anonymous surveys.

3.1 Student Participation

Instructor observations and student peer reviews reported fewer complaints about team members not carrying their share of the project load as compared to previous semesters of these courses. In addition, the instructor observed that class attendance and enthusiasm was higher than in previous semesters. Thus, the use of the technology provided by the HP grant appears to have had a positive impact on students' participation.

3.2 Communication and Engagement

The following questions were asked in an anonymous survey of the students in the courses impacted by the grant. A four-point scale was used (3 = very often, 2 = often, 1 = sometimes, 0 = never):

- Participate: For the days you attended class, how often did you ask questions in class or contribute to class discussions?
- Communicate: For the days that you attended class, how often did you communicate with other students during class time about the topics addressed in that day's class?
- Engaged: For the days you attended class, how often were you on-task and actively engaged in learning?

Figure 1 gives the mean results for these questions for the students in the sections impacted by the grant. The results indicate that students in these sections felt that their participation, communication with other students, and engagement was very high.

	Participate		Communicate		Engaged	
	DB	**SW Engr**	**DB**	**SW Engr**	**DB**	**SW Engr**
Mean Value	2.3	2.5	2.0	3.0	3.0	3.0

Figure 1. Participation, Communication, and Engagement reported by Students Impacted by the Study (Scale: 3 = very often, …, 0 = never)

3.3 Quality of Student Work

To assess the quality of the students' final products for each course, the instructor compared the final products with those submitted by previous semesters' students. In the instructor's opinion, the final group projects for the sections impacted by the HP project grant were more complete and of higher quality than in previous semesters. The students also did a better job of working together to identify and mitigate risks caused by schedule constraints, lack of experience of the team, and the need to narrow the requirements for the first increment to yield a small working prototype that could be of use to the customer and would be doable within a single semester. Thus, the use of the technology provided by the grant appears to have had a positive impact on the quality of the students' final products.

3.4 Student satisfaction

At the end of each semester, students in every course at FSC are asked to anonymously evaluate the course and instructor. A 5-point scale was used for each question (5= Excellent, …, 1 = Poor.) The results from the following questions from the anonymous student evaluations were analyzed for this study because we consider them to be the most relevant to student learning:

- Relevance: "Instructor's ability to make the material relevant to the course."
- Challenge: "Instructor's ability to make the course intellectually challenging."
- Accomplishment: "Instructor's ability to foster feelings of accomplishment."

Figure 2 shows that the course sections that included a collaborative service learning component (SL) along with collaboration technology(CT) were rated very high by the students in each of these three areas. Note that the service learning sections of the courses were rated higher than the sections that did not use service learning, regardless of whether collaboration technology was used with the service learning course. This result should help to allay any fears of instructors who are concerned about a potential negative impact on faculty evaluations when several technology changes are introduced in a course in a single semester with little time for instruction on how to use the technology.

	Relevance		Challenge		Accomplishment	
	DB	**SW Engr**	**DB**	**SW Engr**	**DB**	**SW Engr**
SL:No. CT:No	4.67	4.50	4.39	4.80	4.61	4.73

SL:Yes. CT:No	4.89	4.80	4.89	4.80	4.67	4.90
SL:Yes. CT:Yes	4.89	4.89	4.84	5.00	4.89	4.78

Figure 2. Impact of Service Learning (SL) and Collaboration Technology (CT) on Student Satisfaction (Scale: 5 = Excellent, …, 1 = Poor)

3.5 Student opinion of collaboration technology

In addition to the survey questions, students impacted by the grant were asked to give anonymous input concerning their impressions of the use of the tablets, collaboration technology, and digital ink during class. All of the responses were positive. The following are some of their responses:

- "The tablets changed everything. They made working together on projects completely easy."
- "During every phase of the project it was helpful to have the tablet computers. The portability of the computers is a major advantage to being tied-down to a desktop."
- "Being able to go across the room and take the tablet computer along to discuss a design option or a coding problem was wonderfully helpful."
- "OneNote was used so the whole class could work on a single project, but we would not have to be huddled over one another."
- "During the design phase of the software project we collaborated using OneNote: drawing diagrams, taking notes, and brainstorming. During the coding phase we used Microsoft office, Internet Explorer, and Eclipse."
- "We were able to use One Note to connect directly to each other and work at the same time to get work done faster and with better communication."
- "I would not have wanted to give up the pen as an input option."
- "There are certain things that the pen is vital for, like drawing a diagram."

4.0 CONCLUSIONS

Due to the limited scope of the study, any conclusions must be considered to be preliminary. However, the results from this study indicate that the use of a collaborative service learning project in an undergraduate database course and an undergraduate software engineering course had a positive impact on student course satisfaction, regardless of whether collaboration technology was available for student use in the classroom. Based on faculty and student impressions, when tablet laptops, wireless communication, and collaboration software were used in the classroom for these two courses, the collaboration technology appeared to have a positive impact on student project participation, a positive impact on student on-task communication and engagement with the course content, and a positive impact on quality of student work products.

REFERENCES

1. Association of American Colleges and Universities, *College Learning for the New Global Century, a Report from National Leadership Council for Liberal Education & America's Promise*, Washington, D.C.: AACU, 2007.
2. Bringle, R.G., Hatcher, J.A., Implementing service learning in higher education, *Journal of Higher Education*, 67, (2), 221-239, 1996.
3. Chickering, A. W., Gamson, Z. F., Seven principles of good practice in undergraduate education, *The American Association for Higher Education Bulletin*, March 1987.
4. Dewey, J., *Experience and Education*, New York: Collier Books, 1938.
5. Madley, G., Freeland, C., Brenner, P, A service learning program for CSE students, *35th ASEE/IEEE Frontiers in Education Conference*, Indianapolis, IN: Oct 19-22, 2005.
6. Kolb, D. *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs: Prentice-Hall, 1984.
7. Olsen, A. L., A service learning project for a software engineering course, *The Journal of Computing Sciences in Colleges*, 24, (2), 130-136, 2008.
8. Scorce, R., Perspectives concerning the utilization of service learning projects for a computer science course, *The Journal of Computing Sciences in Colleges*, 25, (3), 75-81, 2010.
9. Traynor, C., McKenna, M., Service learning models connecting computer science to the community, *Inroads - The SIGCSE Bulletin*, 35, (4), 43-46, 2003.
10. Wilson, D, Experiences of engineering students in post-Katrina service learning programs, *38th ASEE/IEEE Frontiers in Education Conference*, Saratoga Springs, NY: IEEE, Oct 22-25, 2008.

PYTHON AND VISUAL LOGIC© - A GOOD COMBINATION FOR CS0*

Krishna K. Agarwal
Department of Computer Science, Louisiana State University in Shreveport
Shreveport, LA 71115, (318) 795-4283, krishna.agarwal@lsus.edu

Achla Agarwal
Division of Business and Computer Science, Bossier Parish Community College
Bossier City, LA 71111, (318) 678-6048, aagarwal@bpcc.edu

Leslie Fife
Department of Computer Science, Louisiana State University in Shreveport
Shreveport, LA 71115, (318) 797-5093, leslie.fife@lsus.edu

ABSTRACT

There are many challenges associated with introductory classes in computer science. In this course, students often get their first taste of computer programming. As part of this introductory course, algorithms, flowcharts, pseudo-code and a high level programming language are often introduced. There can be some concern over the selection of each of these items as we introduce new students to our discipline. This paper recommends Python with Visual Logic© for teaching CS0, the introductory programming course for computer science majors. Python is a full featured, high level programming language. Visual Logic© is a tool which allows the students to create executable flowcharts. These features make Python and Visual Logic© suitable for CS0 and are introduced in parallel.

INTRODUCTION

The CS0 course provides an introduction to problem solving and the use of algorithms using flowcharts and programming in a high level language [1, 3]. This is a required course for all students with no programming experience. The use of Python with

---

Visual Logic© is not a new recommendation [7]. However, rather than using Visual Logic© for the first part of the course and then introducing Python as in [7], we integrate both tools throughout the course.

Visual Logic© (www.visuallogic.org) is a program which allows you to create and execute flowcharts [4]. However, including the use of a higher-level programming language gives the students a firmer foundation and a way of checking if the logic in their algorithms and flowcharts is correct.

Python (www.python.org) is a modern, higher-level language available since 1991 when it was invented by Guido Van Rossum [5]. It runs under Windows, Linux and a variety of other operating systems. It is available at no cost with extensive language support for the language available. The simple, pseudocode like syntax of Python makes transition from algorithm description to code easier for the beginning students [5, 10]. The move to Python has occurred at the CS0 level or higher in several school [2, 6, 9, 10, 11]. There is a large community that supports Python and many excellent books [5, 8].

Washington and Lee University report the transition to Python throughout their curriculum [10]. Michigan State introduced Python through CS1, followed by C++ in CS 2 [6]. They found no performance differences in CS 2 based on the language used in CS 1 (C++ or Python) [6]. They also reported no deficiencies in CS2 based on the language choice in CS 1 [6]. Zelle points out that a scripting language is ideal for the type of programming done in CS0, CS1 and CS2 courses [12]. Additionally, with support for both modular and Object-Oriented programming, Python provides an appropriate tool when objects are important early in the curriculum [12]

Python presents many advantages for CS0. Among these are that variables do not have to be declared (they have dynamic typing), automatic memory management, and modularity [1, 2, 3]. These advantages make Python suitable for both small and large scale programming. Python also has a large and comprehensive standard library which includes access to a variety of GUI toolkits [1, 2, 3]. Python also supports object-oriented programming and provides a good interface with C/C++, Java and other languages.

The rest of this paper is devoted to demonstrating a simple example of an algorithm and its implementation in Python and Visual Logic© in a CS0 course. Review of the examples and tutorials at www.python.org is also recommended. This website also lists many advantages of Python for introductory programming.

A SIMPLE PROBLEM

Consider the following problem which is typical of that assigned in a CS0 course. We will show the algorithm and provide a Visual Logic© flowchart as well as Python code. A list of employees names, hours worked in a week and the hourly rate of pay are entered. The program must read in name, hours worked and pay rate. The program must calculate the pay for each employee, paying time and a half (1.5 times pay) for any hours over 40. The program must print the amount of pay for each employee and the total amount of the payroll. Some sample data with the corresponding output is shown below.

```
Input:                                  Output:
"John Smith", 50, 10                    John Smith's pay=$550.00
"Jane Doe", 40, 15                      Jane Doe's pay=$600.00
"Mark Manning", 60, 12                  Mark Manning's pay=$840.00
"zzzz"                                  Total pay = $1990.00
```

The input shown provides the ability for the instructor to have a discussion on marking or detecting the end of input when processing data and could include flags, or end of file indicators. This makes the introduction of Boolean controlled loops very natural with an example that students will readily understand.

This simple problem involves Boolean controlled loops, selection on hours worked and either console or file I/O.

A simple algorithm for this problem may be as follows:

```
WHILE employees available for processing:
   Read EMPLOYEENAME, HOURSWORKED, and PAYRATE
   //Calculate the pay for the employee
   IF HOURSWORKED > 40
      PAY = PAYRATE*40 + 1.5*PAYRATE*(HOURSWORKED-40)
   ELSE
      PAY = PAYRATE*40
   Print the EMPLOYEENAME and PAY
   Add PAY to the TOTALPAYROLL
END WHILE

Print TOTALPAYROLL
```

A beginning student should be able to develop this algorithm, possibly with some help if early in the course. However, there is nothing particularly challenging about this algorithm either. If the algorithm was already developed, a student should be able to follow the logic and work it out on paper if given appropriate input.

We then develop the Visual Logic© flowchart, Python code or both. The flowchart is executable. So, the student can run the sample data and compare it to the sample output. The Python code would also be executable.

Figure 1 demonstrates a simple Visual Logic© flowchart which calculates an employee's pay given the rate of pay and number of hours worked. It also produces the sum of all the employees pay. A mark in the input or output box indicates that the input or output should be done using a file. The filename is entered into the corresponding input or output box in the flowchart by double clicking and entering the file name. A built-in formatcurrency() function provides output formatting.

Figure 2 gives the corresponding Python program. Comments begin with a '#'. There is no need to declare variables, making it like Visual Logic©. The print statement includes a command "{0:.2f}" to format the number stored in the variable pay. The Python program is largely self-explanatory. Note that all statements belonging to then and else parts of the if statement in a Python program must be indented. There is no need for begin, end or any other symbols to indicate a block of statements. Indentation of all statements inside the while loop is necessary in the Python program.

Figure 1 - Visual Logic© Program

```
#This program calculates an employee's pay given the rate of pay and
#the number of hours the employee has worked. A loop permits processing
#several employees. The total pay is also printed.

total_pay = 0
name = input("Enter name of employee:")

while name != "zzzz":

  hours_worked = float(input("Enter the hours worked:"))
  rate_of_pay = float(input("Enter rate of pay:"))

  if hours_worked>40:
    pay = rate_of_pay * 40 + 1.5 * rate_of_pay * (hours_worked - 40)
  else:
    pay = rate_of_pay * hours_worked

  total_pay = total_pay + pay
  print ("The pay is:${0:.2f}".format(pay))
  name = input("Enter name of employee:")

print ("Total pay=${0:.2f}".format(total_pay))
```

Figure 2 - Python Program

CONCLUSIONS

Several studies indicate the benefits of using Python in CS0 and other courses in the computer science curriculum. The pairing of Visual Logic© with Python creates a set of mutually supporting tools that allow students to quickly implement algorithms and test problem solutions. Visual Logic© and Python provide a strong choice for CS0, because the syntax of the programs are significantly simpler. Not only are the Python programs shorter in length, they are clearer to beginning students of computer science because they are a closer to pseudocode and are missing some of the burden required by many typical languages, such as C, C++ and Java. For example, code for input and output is quite simple.

Although not discussed in this paper, there are many features of Python that make it suitable for CS1, CS2 and other advanced courses in computer science [2]. These include support for interactive code development, support for multiple programming paradigms (structured, object-oriented, functional, aspect-oriented, etc.), extensibility, and easy integration with other languages (C, C++, Java, etc.).

REFERENCES

1. Agarwal, K., Agarwal, A., Python puts a squeeze on Java for CS0 and beyond, *The Journal of Computing Sciences in Colleges*, 23(6), June 2008.

2. Agarwal, K., Agarwal, A., Python for CS1, CS2 and Beyond, *The Journal of Computing Sciences in Colleges,* 20(4), April 2005.

3. Agarwal, K., Agarwal, A., Simply Python for CS0, *The Journal of Computing Sciences in Colleges,* 21(4), April 2006.

4. Crews, T., and C. Murphy, *A Guide to Working With Visual Logic*, August 2008.

5. Deitel, H., Deitel, P., Liperi, J., and B. Wiedermann, *Python: How to Program*, Upper Saddle River, CT: Prentice Hall, 2002.

6. B. Enbody, R., Punch, W., and M. McCullen, Python CS 1 as Preparation for C++ CS2, *40th SIGCSE Technical Symposium on Computer Science Education*, 41(1), 116-120, 2009.

7. Gudmundsen, D., Olivieri, L., Sarawagi, N., Using Visual Logic: Three different approaches in different courses - General Education, CS0, and CS1, *The Journal of Computing in Small Colleges*, 26(6), June 2011.

8. Harms, D., McDonald, K., *The Quick Python Book*, Greenwich, CT: Manning, 2000.

9. Miller, B., Ranum, D., Freedom to Succeed: A Three Course Introductory Sequence using Python and Java, *The Journal of Computing in Small Colleges*, 22(1), 106-116, 2006.

10. Necaise, R., Transitioning from Java to Python in CS 2, *The Journal of Computing Sciences in Colleges*, 24(2), 92-97, 2008.

11. Shannon, C., Another Breadth-first approach to CS I using Python, *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 35(1), 248-251, 2003.

12. Zelle, J., *Python Programming: An Introduction to Computer Science*, Wilsonville, OR: Franklin, Beedle & Associates, Inc, 2003.

Appendix A - Sample Problems

1. Jimmy works at the local air force base. On Wednesday, Jimmy worked from 8:12 hours until 16:38 hours with a lunch break from 12:02 hours until 12:24 hours. Calculate how long Jimmy worked on Wednesday.

2. A monkey is being fed some food. Read in the starting weight in lbs:ozs. Also read in the ending weight in lbs:ozs (you may assume this is smaller than the starting weight. Find the difference and print out the amount of food consumed by the monkey in lbs:ozs. Hints: convert all values to ounces first. Use the "find" command to locate the ":" in the input data.

3. You want to invest your money in a local bank. They are currently offering several plans, one with simple interest and the other with compound interest. The rates offered at the bank are 5%, 10% and 15%. Your task is to vary the principle invested from $10,000 to $15,000 with an increment of $1,000 for each rate.

4. Print the first 10 Fibonacci numbers in reverse order. Note that it is easy to use an array to store the numbers in ascending order and then print the array in reverse. The Fibonacci series is as follows:

   1, 1, 2, 3, 5, 8 ……

   The first two numbers are 1. All other numbers are the sum of the previous two.

MULTI-CORE PROGRAMMING IN JAVA*

CONFERENCE TUTORIAL

Timothy J. McGuire
Department of Computer Science
Sam Houston State University
Huntsville, TX 77341-2090
936-294-1571
mcguire@shsu.edu

ABSTRACT

Multi-core computers are now ubiquitous. In order to make use of the full power of these systems, it is now imperative that we teach parallel programming techniques to all students. Despite the fact that the high-performance computing community has been programming parallel applications for many years, most programmers have only a cursory understanding of the issues involved in developing multi-core applications. As machines with 32 or more cores become commonplace, students must gain a working knowledge of how to develop parallel programs. It seems evident that students must be exposed to concurrency throughout the curriculum, beginning with the introductory CS sequence. Parallel processing can no longer be relegated to an upper-level elective, but it must be integrated in the entire curriculum. Most parallel programming courses are taught using MPI or OpenMP in C or Fortran. Since many students are learning Java as their principal language, it is desirable to introduce students to parallelism using that language. Java threads have the advantage of being a part of the language, making it useful for parallel programming on multi-core machines. Nonetheless, even though threads may be seen as a small syntactic extension to sequential processing, as a computational model, they are non-deterministic, and the programmer's task when using them is to "prune" that non-determinism.

For C and Fortran, OpenMP allows for a higher level of abstraction, allowing a programmer to partition a program into serial regions and parallel regions (rather than a set of concurrently executing threads.) It also provides intuitive synchronization constructs. These help eliminate many of the pitfalls of non-determinism.

For Java, similar mechanisms such as PJ, JOMP, and JaMP have been proposed. This tutorial will survey the parallel programming landscape, summarize how the OpenMP approach to multi-threading can be applied to Java, and illustrate how it can be used to introduce parallelism into the curriculum for novice and intermediate programming students.

ACKNOWLEDGMENTS

The author wishes to thank Dr. Michael Scherger of Texas A&M University–Corpus Christi for his assistance and advice in developing this tutorial.

DESIGNING AND IMPLEMENTING UNSUPERVISED ONLINE DATABASE LABS*

Gang Qian
Department of Computer Science
University of Central Oklahoma
Edmond, OK 73072
405 974-5716
gqian@uco.edu

ABSTRACT

Lab-based learning that provides students with hands-on experience is an integral component of Computer Science (CS) education. Unfortunately, CS departments in many small institutions often lack the resources necessary to offer a full spectrum of lab courses in their curriculum. In this article, we present the setup and contents of a sequence of unsupervised online labs for a typical undergraduate database course. Problems that occurred during the implementation of the labs and how they were addressed are also discussed. The labs were designed to be fully self-explanatory with no need for teaching assistants, extra credit hours, or physical laboratories. The labs incorporate a tutorial style which allows students to learn new technologies outside of the classroom and apply them to a comprehensive course project. Multiple years of feedbacks indicated that the labs were effective in helping students learn and preparing them for real-world applications. Our experience shows that unsupervised labs, if properly designed and administered, can be a promising alternative to physical labs in CS education.

1 INTRODUCTION

Computer Science (CS) is a discipline of applied science in which practice is as important as theory. College graduates with CS degrees are expected to be directly employable in the IT industry. Therefore, every viable CS program must incorporate a lab component that helps its students practice knowledge learned from lectures. In the

---

CS education literature, there are a number of papers that discuss the importance of lab-based learning for students, such as [1] and [2].

Unlike their counterparts in large research-oriented institutions, CS departments in smaller or primarily undergraduate institutions (PUIs) often have difficulties in offering a full range of lab courses due to insufficient resources. This can be a three-fold problem, including credit hour limitation, lack of graduate students, and funding shortages. Firstly, many PUIs have a cap on the total number of credit hours that a student can take before graduation and emphasize a liberal arts core curriculum that encompasses a large number of general education courses. As a result, the number of credit hours that can be used for CS major courses is limited and it is difficult to add more lab hours into a CS program. Secondly, CS departments in PUIs usually have no or small graduate programs that cannot provide enough quality teaching assistants for the lab courses. Thirdly, funding shortages often make it hard to offer new lab courses in PUIs. The recent economy downturn has caused budget cuts in virtually all higher education institutions. Colleges and universities scrutinize every request for a new course and it becomes even less likely than before that new lab courses can be added into a CS curriculum.

The issues discussed above demand that CS faculty members in PUIs find quality alternatives to physical CS lab courses. In this paper, we present a sequence of unsupervised online labs for a typical undergraduate database course. We developed the labs as an alternative to physical database labs. Because of their intended unsupervised use, the labs were designed to be fully self-explanatory and easy to follow. After multiple years of implementation of the labs, we received very positive feedbacks from a number of students who have taken the course. It is our conclusion that unsupervised online labs, if properly designed and implemented, can be as effective as physical labs in providing students with hands-on practices of classroom knowledge.

The rest of the paper is organized as follows. Section 2 presents the details of the unsupervised lab sequence. We cover both the setup and the contents of the labs. In Section 3, we discuss student feedbacks as well as problems that we faced and addressed while administering the labs in our undergraduate database course. Conclusions and future works are given in Section 4.

2 THE UNSUPERVISED DATABASE LABS

In this section, we provide the details of the unsupervised database lab sequence. We first introduce the setup of the hardware/software environment.

2.1 Online Lab Environment Setup

Because there is no physical lab, the environment is set up such that students can access the database system through the Internet. Figure 1 shows the architecture of the lab environment. This is a typical client-server architecture. Student computers access the lab servers as clients by logging onto the gateway server over the Internet. Through the gateway server, students have access to the database, Web and application servers. Most lab works can be completed on the database server. Students need to use the Web and application servers for labs that are related to a course project on Web-based database application development.

In order to reduce cost, we chose mostly mainstream open-source software for the servers and clients. Table 1 provides a summary of software used in the setup.

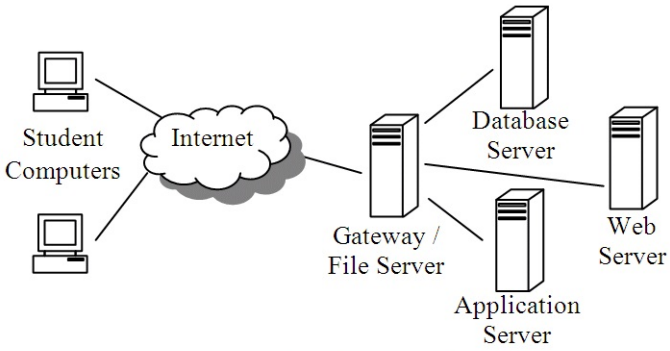


Figure 1 Lab Hardware Architecture

Table 1 Software Used in the Lab Setup

Gateway/File	Database	Web	Application	Client
Ubuntu Linux	Oracle	Apache HTTP Server	PHP	PuTTY / WinSCP

Note that we adopted Oracle for the database server because of its large user base in the industry. Its educational license fee is quite affordable at $500 a year [4]. A good alternative is the free MySQL database system, which is a component of the popular LAMP [3] open source software bundle. On the client side, students use PuTTY, a free terminal emulator, to access the servers. A free SFTP utility (WinSCP) is also used to transfer files between the servers and student computers.

2.2 The Database Lab Sequence

The lab sequence contains 11 assignments, which cover database topics including ER/EER models, SQL, views, transactions, stored procedures, and triggers. For each assignment, schema and data files are provided so that students can easily create the tables used in the lab. We use standard SQL wherever possible to maintain generalness. Oracle-specific topics such as SQL*Plus and PL/SQL are also covered. In preparation of a course project, we have one lab on PHP database application development.

To enhance learning effectiveness, we adopted an interactive teaching style in the labs. After completing an operation, students are asked to check what has happened on the screen and write down their observation. For many operations, they are also asked to explain what they see. In this way, students cannot simply follow the instructions, complete the lab, and forget about it a moment later. They have to think to successfully complete a lab.

One important component of the lab sequence is to introduce students to reference manuals on SQL and other database technologies. Students can learn more from the manuals, if they are interested in topics beyond the labs. Due to our use of the Oracle database, we present the Oracle OTN website, which contains a number of reference books on various database topics.

A detailed discussion of the labs is given as follows.

2.2.1 Lab 1: Introduction

For many students, Lab 1 is the first time that they use a full relational database system. The objective is to familiarize them with command line-based access and basic operations. In the lab, students connect to the database through SQL*Plus, the Oracle command-line client utility. Students are then asked to complete a series of basic SQL commands, including create table, insert, select, delete, commit and drop table. SQL*Plus command DESC is also explained and used in the lab.

2.2.2 Labs 2 and 3: ER/EER Model

Labs 2 & 3 are "drill" labs because they do not involve database operations. Lab 2 provides data requirements and asks students to build an ER model based on the requirements. Also covered is the mapping of the completed ER model to the corresponding relational model. The lab focuses on areas of common student mistakes, such as weak entity types, recursive relationship types, ternary relationship types, relationship attributes, and multi-valued attributes.

Lab 3 extends Lab 2 by adding object-oriented data requirements. Students modify the ER model in Lab 2 into an EER model that covers the new OO requirements and map it into the corresponding relational model. Areas of focus in Lab 3 are IS-A relationships, local attributes, generalization, and multiple inheritance.

2.2.3 Lab 4: Basic DDL and DML

In Lab 4, we cover basic DDL and DML statements. Students are asked to implement the database schema derived from Lab 3. The emphasis is placed on defining table constraints and testing them using insertion, deletion and update statements. Special situations, such as defining tables that reference each other and inserting records into such tables, are also discussed in the lab.

2.2.4 Labs 5 and 6: Relational Algebra, SQL Queries and Advanced SQL

Lab 5 covers basic SQL queries. For each query requirement, students write a relational algebraic expression and the corresponding SQL query statement. Only basic relational operations are included in this lab, such as SELECT, PROJECT, CROSS PRODUCT, and JOIN.

The contents of Lab 6 are based on the same database schema used in Lab 5, which has been created in Lab 4. Advanced SQL queries are required in this lab, including outer join, theta join, and aggregated functions with GROUP BY and HAVING. We emphasize the use of correlated select statements in implementing queries that involve set minus, division and other complex relational operations. Advanced DML statements with embedded queries are also covered. In both Labs 5 and 6, we ask students to use a structured divide-and-conquer methodology to construct the relational algebraic expressions and the SQL statements.

2.2.5 Lab 7: Views

Lab 7 focuses on view operations. We cover both view creation and querying. Materialized views and data manipulation of base tables through views are also discussed in the lab.

2.2.6 Lab 8: Web-based Database Application Development in PHP

Lab 8 is one of the longest labs in the lab sequence. It teaches basic Web-based database application development in PHP. It covers PHP language structures embedded in HTML, value-passing between PHP pages, PHP database connections, and a generic session management scheme. Cursor, an important database programming concept, is also introduced in this lab.

2.2.7 Lab 9: Stored Procedures in PL/SQL

We cover stored procedures in Lab 9. The lab first introduces the basic elements of PL/SQL, the Oracle database programming language. It then discusses the syntax of creating a stored procedure and programming in PL/SQL. Procedure invocation, parameter passing and debugging in both SQL*Plus and PHP are also covered in the lab.

2.2.8 Lab 10: Transaction Processing and Concurrency Control

Lab 10 is another long lab in the lab sequence. In this lab, students are asked to open two terminal windows to simulate two database users entering database commands concurrently. Students observe the effects of various mechanisms employed by database systems to ensure data integrity in a multi-user environment, including the commit and rollback statements, write locks, deadlock resolution, and ANSI isolation levels. The concept and handling of long-duration transactions are also discussed.

2.2.9 Lab 11: Database Triggers

The last lab in the sequence covers database triggers. Topics include statement and row-level triggers, and nested and recursive triggers. An emphasis is placed on helping students understand the mutating table error and how to avoid the error.

The labs in the sequence are not independent from each other. Figure 2 shows the dependency structure of the labs. It is worth noting that the lab sequence has a bias toward database application development. Due to students' workload consideration, physical database implementation topics, such as indexing and query optimization, are not covered in the labs.

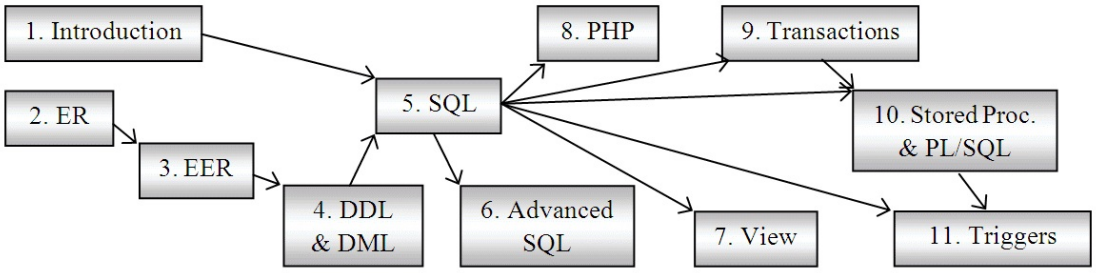


Figure 2 Dependency Structure of the Lab Sequence

3 FEEDBACKS AND PROBLEMS IN USING THE LABS

In this section, we discuss student feedbacks and the problems that we faced in implementing the lab sequence.

3.1 Students' Positive Feedbacks

The lab sequence was first introduced in summer 2006. With small improvements overtime, it has since been used for more than ten semesters. We received overwhelmingly positive feedbacks from students and alumni, which can be summarized into the following three points:

1) The labs helped students understand and practice what was taught in class;
2) Students were able to learn a real-world mainstream database system and the basics of database application development;
3) The labs helped students obtain employment after graduation.

3.2 Problems in Using the Labs

The lab sequence inevitably had issues that needed to be addressed. When the labs were first introduced as an unsupervised learning instrument, we did receive complaints for lack of help in the first few semesters. We updated the lab write-up by emphasizing a tutorial style. We also strengthened other help channels, such as prompt email correspondence, well-trained tutors, and a dedicated online discussion board. The issue was quickly resolved and we received no complaint on lack of help in the following years.

Another student complaint was about the workload. In addition to the labs, we also require multiple written assignments and a course project. At first, we believed that students would learn course materials better through more practices and thus a heavy workload was justified. We quickly realized that the heavy workload did not necessarily lead to better knowledge retainment. CS seniors often take multiple major courses with non-trivial assignments. If they spent too much time on one course, their performance in other courses will suffer. To solve the problem, we allowed more time for each lab assignment. We also designated a few labs as optional. Lab accounts were kept for at least one month after the end of a semester so that students could still try the optional labs if they were not able to finish them during the semester. These changes were effective as we no longer received complaints on workload after the changes had been implemented.

Some students raised the issue that, since PHP and PL/SQL were never formally taught in class, it was not fair to require them in the labs and the course project. We explained that, as senior-level students, they should have the ability to learn new programming languages by themselves. To further help students, we refined our PHP and PL/SQL labs by making them more self-explanatory. We also gave a brief discussion on key elements of PHP and PL/SQL in class.

Since the labs were to be re-used in multiple semesters, one concern that we had was about cheating. To reduce the possibility, in addition to repeatedly emphasizing the importance of academic honesty, we implemented a few supplementary administration policies. We grade the labs based an all-or-nothing criterion, that is, as long as a student completes the lab with a reasonable amount of effort, even if there are errors in the

submission, we will still assign a full score to the student. We also changed the labs into group assignments and encouraged students to discuss the labs in their study group and work on the solutions together. Although there was no guarantee that cheating could be totally eliminated, the policy changes did encourage the majority of the students to work by themselves and thus benefit from the labs.

There were other issues that occurred over the years. For example, a server crashed during a weekend. To improve the availability of the lab environment, we set up the servers such that they can automatically restart every day. Student accounts are set up in a way that passwords cannot be changed by students. If a student has some technical difficulty, the instructor can log into the student's account and look into the problem.

After multiple semesters of improvement, the lab sequence is much refined now. We receive very few if any complaints every semester. Students' feedbacks clearly indicate that the labs were successful in helping them learn and practice database technologies.

4 CONCLUSIONS

Although there is an ongoing debate on close labs versus open labs in CS education [5], to the best of our knowledge and literature survey, we are the first to propose the idea of using structured unsupervised labs in place of more costly physical labs. Our positive experience shows that, if properly designed and implemented, unsupervised labs can be a quality alternative to dedicated lab courses.

In future work, we plan to add a few more components into the lab sequence, including database security and cloud-based database applications. We also plan to apply the idea to other CS courses where labs are desired but cannot be afforded. Since different courses have different types of topics and focuses, the challenge will be designing labs that are suitable for those courses.

REFERENCES

1. Hazzan, O., Lapidot, T., Ragonis, N., Hazzan, O., Lapidot, T., Ragonis, N., Lab-Based Teaching (Book Chapter), *Guide to Teaching Computer Science*, Springer London, 2011.

2. Knox, D., Wolz, U., Joyce, D., Koffman, E., Krone, J., Laribi, A., Myers, J. P., Proulx, V. K., Reek, K. A., Use of laboratories in Computer Science education: Guidelines for good practice, *Proceedings of the 1st Conference on Integrating Technology into Computer Science Education*, 167-181, 1996.

3. Lee, J., Brent W., *Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP*, Addison Wesley, 2002.

4. Oracle Academy, https://oai.oracle.com/en/program_guide.html.

5. Kumar, A., Closed labs in computer science I revisited in the context of online testing, *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 539-543, 2010.

TEACHING COMPUTER ARCHITECTURE THROUGH SIMULATION (A BRIEF EVALUATION OF CPU SIMULATORS)*

Timothy Stanley, PhD Computer and Network Sciences, Utah Valley University, Orem, Utah 84058, 801 863-8978, TStanley@uvu.edu

Vasu Chetty, Matthew Styles, Shin-Young Jung, Fabricio Duarte, Tin-Wai Joseph Lee, Michael Gunter, Computer and Information Sciences, Brigham Young University - Hawaii

Leslie Fife, PhD Dept. of Computer Science, LSUS, Shreveport, LA 71115

ABSTRACT

In our computer architecture class, we researched logic simulators for design of educational computers. We have regularly had students design, create and operate a simulated computer as part of this course. This has allowed students to understand the internal details of instruction decoding and data path control. In the past we have used Multimedia Logic (MML). MML is open source and free and it has an attractive user interface. However, a number of improvements have been made to Logisim and we felt a re-evaluation was appropriate. The machine we chose to build was Linda Null's sixteen bit MARIE computer. We implemented this computer in Multimedia Logic, Logisim, Cedar Logic and CPU Sim. The implementation in CPU Sim allowed us to add and test additional instructions to the MARIE instruction set and to use CPU Sim as an assembler for our other designs. This paper compares student designs and discusses the pedagogical value and ease of implementation of these designs.

INTRODUCTION

To fully appreciate how microcomputers work, one needs to understand how an instruction decoder modifies the data path in the processor to accomplish the instructions in its architecture. Our experience is that this understanding and appreciation comes best through designing and implementing the data path and control circuitry of a custom processor design. This could be done using logic circuits in a lab, but we believe this approach is both unnecessarily expensive and needlessly complicated within the computer science curriculum.

There are several good open source logic simulation tools available, and you can use one of these to build the data path and control circuitry [1]. This avoids an expensive lab, broken parts, understanding how logic devices relate to the circuit inputs and outputs, and additional complications.

Our most recent computer architecture class was small, six students, but they were all excellent students. As is customary, we had a processor design project. We chose to design, layout and build the MARIE computer from the book *Essentials of Computer Organization and Architecture* by Null and Lobur [2]. MARIE is an acronym meaning "A Machine Architecture that is Really Intuitive and Easy. One reason for this choice was that all students were familiar with MARIE from their computer organization class.

One problem in using MARIE is the absence of indirect load and indirect save instructions. The indirect load can be provided by using two instructions: an accumulator clear followed by an indirect add. The only way we found to indirectly save was to perform a math operation on the save instruction, effectively modifying the code. Self-modifying code, while elegant is not good programming practice, so we chose to add an indirect load and an indirect save to the thirteen instructions defined by Null and Lobur.

The MARIE architecture is a Von-Neumann architecture consisting of a program counter, a memory address register, a memory data register, an instruction register, an accumulator, an input register, and an output register. Each register is sixteen bits except for the address register which is twelve bits, and the input/output registers which are eight bits. The thirteen defined instructions include add, add indirect, load, store, subtract, input, output, skip conditional, jump, store and jump, jump indirect, clear accumulator, and halt. Since four bits are allocated for operation codes a total of sixteen instructions could be defined.

During this course we also wanted to compare several simulation tools. The class was divided into four teams. One team of two designed and built MARIE in Cedar Logic and another team of two designed and built MARIE using Logisim. One student implemented MARIE in CPU Sim. The last student took an implementation of MARIE from a previous class designed in MML [6], and fixed the bugs in that implementation, made improvements to the I/O circuitry, and implemented the two additional instructions described. This earlier implementation of MARIE was described in [7]. Only portions of each design are shown.

IMPLEMENTATION IN CPU Sim

CPU Sim is a Java application written by Dale Skrien, which allows users to design a simple computer CPU at the microcode level and to run machine or assemble language

programs on those CPU's through simulation [5]. The microcode level is where register transfers are defined or incrementing of the program counter occurs. These register transfer instructions are used to make up the machine instructions that comprise the instruction set architecture ISA of the CPU. While you need a complete understanding of the ISA to build the CPU in CPU Sim, you do not need to construct the data path or control circuitry to implement in CPU Sim. The CPU Sim implementation does not meet all of the goals for the design project, but it provided an assembler to use with the other implementations.

The first step in implementing MARIE in CPU sim is to create and name the hardware modules such as the registers and main memory. The next step is to create the register transfer instructions (called micro code in CPU Sim). The next step is to define the machines instruction fetch sequence. Finally the instructions are formed from the fetch sequence and the micro instructions, completing the definition of the machine. After the machine is defined it can be used to assemble instructions and test execution. Figure 1 shows the MARIE computer in CPUSim.

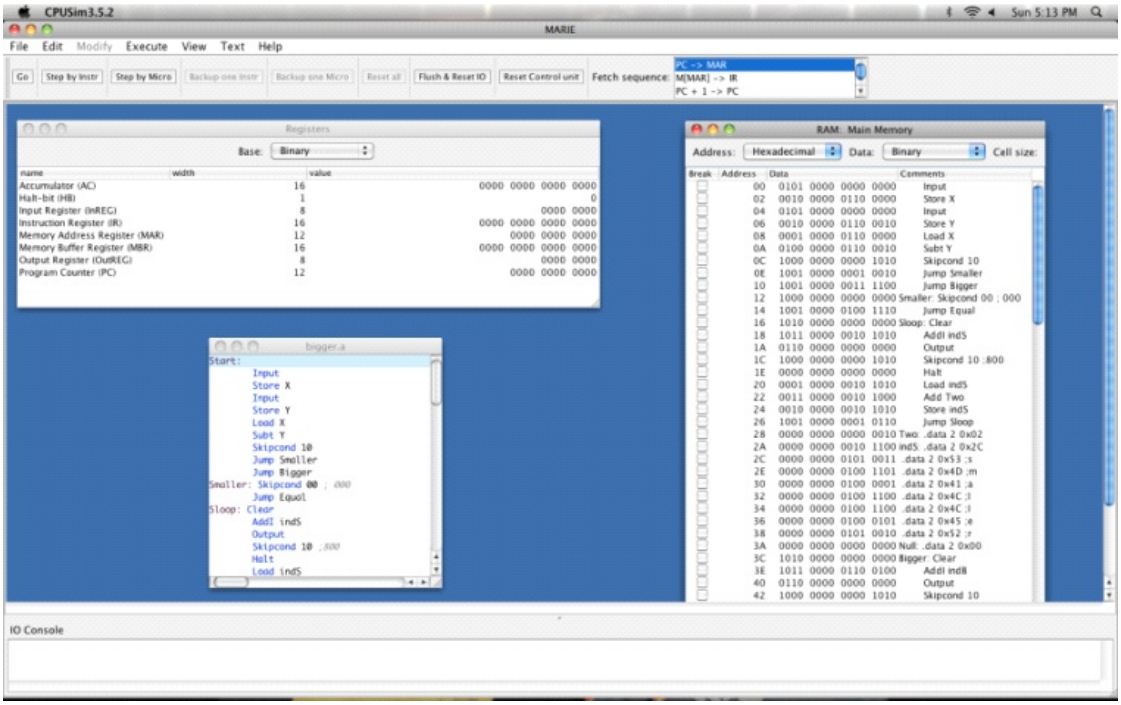


Figure 1 - MARIE in CPU Sim

IMPLEMENTATION IN LOGISIM

Recent Logisim implementations now include a clock, keyboard, and terminal. Logisim provides a great deal of flexibility in device configuration, wire bundles, and sub circuits [4]. Logisim also has the capability to generate logic circuits from truth tables. This capability was used to generate the instruction decoding circuitry. Logisim has the ability to define wires as multiple bit busses. Logisim has the ability to define and use custom circuits. Also, during simulation, Logisim updates memory and registers in real-time while the simulation is running. These features make for a single page machine that demonstrates very well. The only limitation found was the lack of "wireless connectors" available in the other simulators.

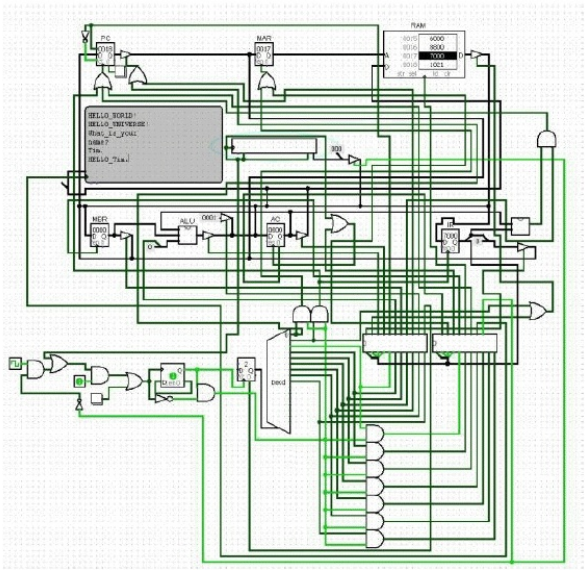


Figure 2 – MARIE in Logisim

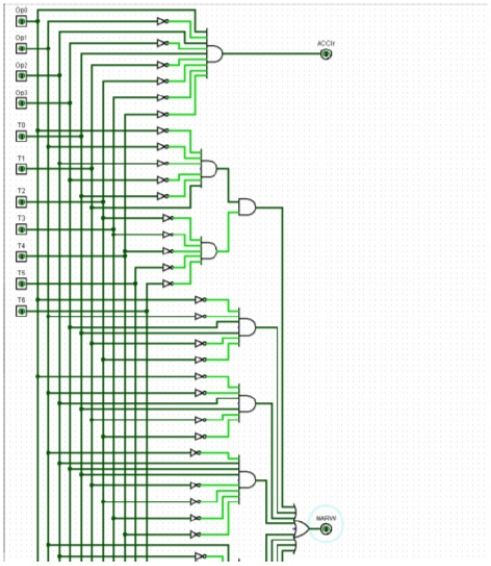


Figure 3 – A portion of the decoding

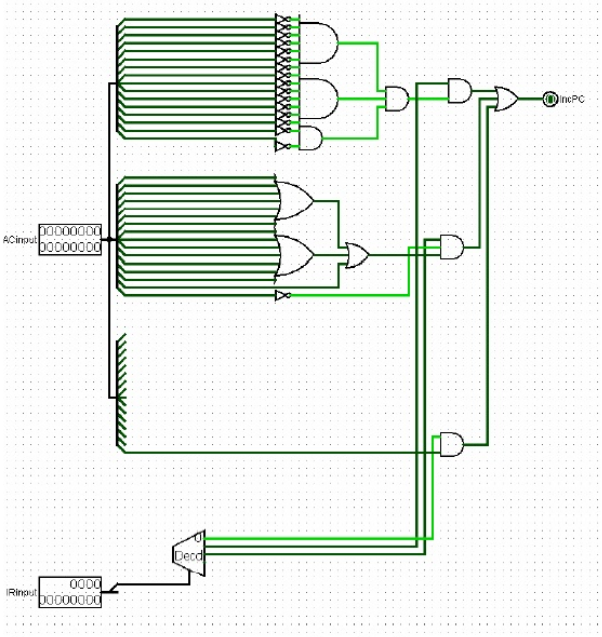


Figure 4 Comparator circuit for "Skip On" instruction in Logisim

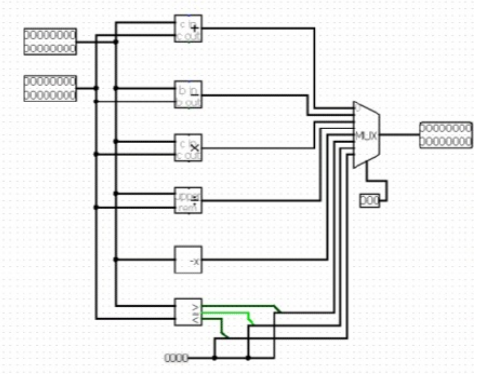


Figure 5 The ALU for MARIE in Logisim

MARIE in Logisim is shown in the Figure 2. The main features shown in this figure include the Program Counter and Memory Address registers followed by the memory at the top of the figure. Below the registers are the ASCII display terminal and the keyboard interface. Below these from left to right are the Memory Buffer Register, the ALU, the Accumulator, and the Instruction Register. To the right of the Instruction Register is a custom circuit that does the comparisons that enable the skip on positive, negative or zero for the value of the accumulator. The balance of the circuitry is the clock and instruction decoder.

As this computer runs, the single bit control wires change color to indicate the logic level they are carrying. The multibit wire do not change color. The registers update their content and memory displays the address and data being accessed. By clicking on the memory, an ascii file can be loaded, giving the memory content. Figure 3 shows a portion of the custom logic circuit produced by the truth table for decoding instructions. Figure 4 shows the comparitor logic circuit. Figure 5 shows the ALU. The ALU defines additional functions beside the add and subtract of MARIE.

IMPLEMENTATION in MULTIMEDIA LOGIC

The original implementation of MARIE in MML was accomplished by Stanley, Prigmore, Mikolyski, Embrey and Fife and was presented in [7]. However, their design was incomplete. They only implemented skip on zero and non zero. So the first goal was to add skip on negative, zero, or positive. This was done by sensing the most significant bit of the accumulator to test for a negative number. The design from 2006 also only included ASCII I/O. To this were added decimal and hexadecimal inputs and outputs. Also added to the Input was an option to specify the input source through a parameter on the input command. Finally, the two new instructions were added. Implementing these required adding to the decoding matrix which was implemented in a read only memory. Figure 6 shows the Instruction Decoder and Control Circuitry for MARIE and Figure 7 shows the I/O screen in MML.

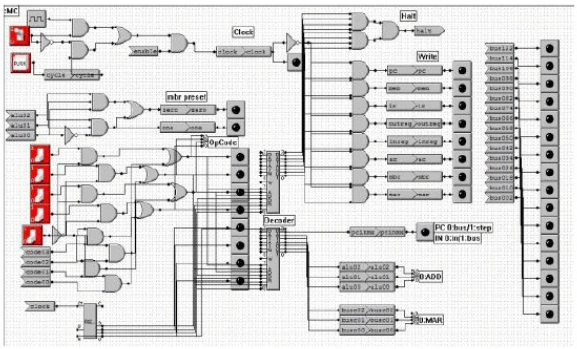


Figure 6 - Instruction decoder and control circuitry for MARIE in MML

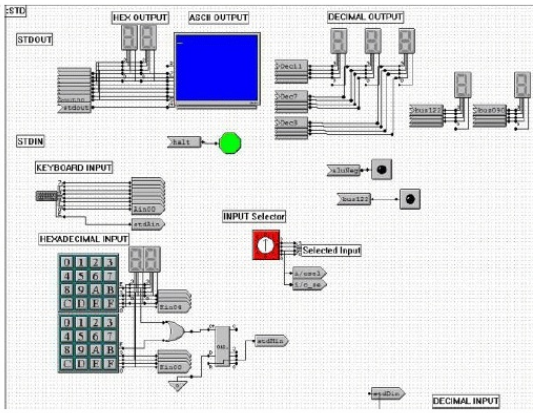


Figure 7 – MARIE I/O screen in MML

IMPLEMENTATION IN CEDAR LOGIC

Cedar Logic has some nice features, particularly for timing investigation and working with a Z80 CPU. But we were frustrated by the absence of any ALU components bigger than a four bit adder or comparator. Also the absence of an ASCII output limited the class of programs that could be directly demonstrated. To overcome this limitation an external program was written to convert hexadecimal into ASCII or Decimal. This limits the ability to demonstrate this design since the output is not real time.

Figure 8 shows the data path in Cedar Logic. This data path is relatively simple to design and layout. Figure 9 shows the decoder and clock circuitry. The complicated part of this circuit is obtaining the contents for the read only memory (ROM). The contents were generated in an excel spread sheet and then exported to the ROM. Figure 10 shows the skip conditional circuitry. Figure 11 is the ALU design.

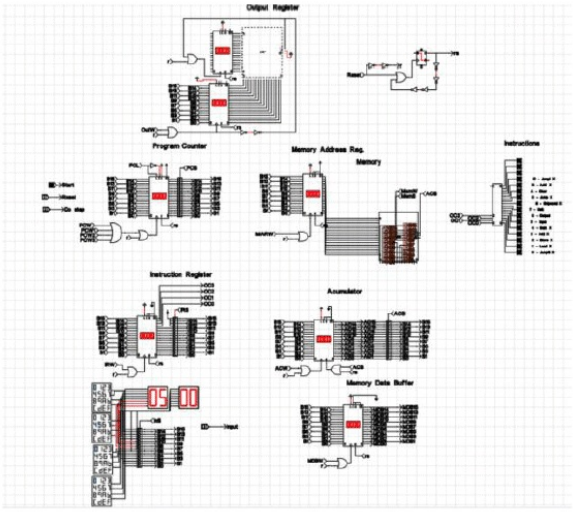


Figure 8 - Data path for MARIE in Cedar Logic

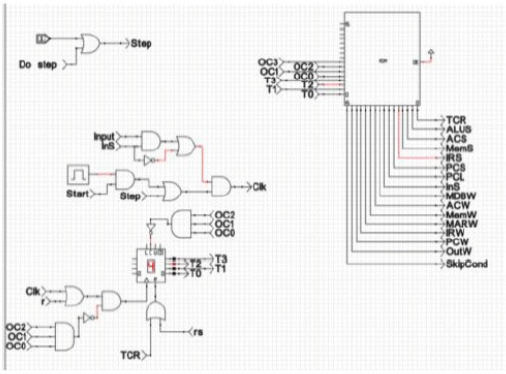


Figure 9 - Instruction decoder and clock for MARIE in Cedar Logic

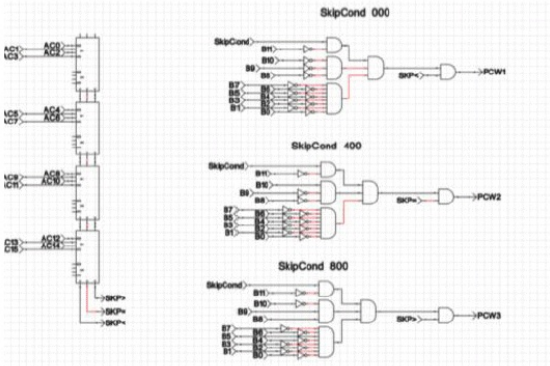


Figure 10 - Skip conditional circuitry for MARIE in Cedar Logic

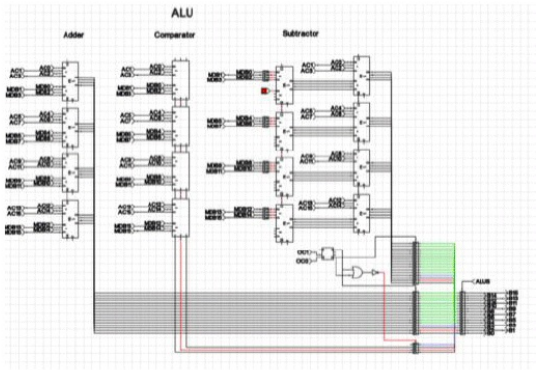


Figure 11 - ALU for MARIE in Cedar Logic

COMPARISONS AND CONCLUSIONS

This was a nice exercise in comparison of various logic simulation tools in the context of a computer architecture class. The least frustration was in the design and implementation of MARIE in Logisim. Logisim fits nicely on one page and the dynamic display of registers and memory allows a very good demonstration of the operation of the CPU. Using only Logisim in this course would be a viable option.

The update of the previously produced MARIE in MML was quite straight forward, but the many pages in the MML layout limits its ability to provide meaningful

demonstrations. Careful layout could likely reduce the number of pages for this design, but not to the level of Cedar Logic or Logisim. The absence of the multi-bit wires in all but Logisim gives a significant advantage to layout in Logisim.

Cedar Logic gave a compact one page data path, and provides a real-time display of the memory contents. Cedar Logic also provides a nice timing facility which is useful in debugging timing issues. But the absence of sixteen bit ALU components and the absence of ASCII I/O capabilities are frustrating.

CPU Sim provides a nice facility to build an emulated machine for a custom instruction set and allows experimentation with microcode, but it does not provide experience with a data path and control circuitry.

These results and conclusions are summarized in the table below.

Table 1. Comparison of Architecture Simulation Tools

Implementation	**Pages**	**Layout time**	**Problems Encountered**	**Pedagogical value**
Multimedia Logic	13	~100 hrs	Problems saving circuits	Limited by the large number of pages
Cedar Logic	4	~40 hrs	No ASCII I/O	Nice timing facility
Logisim	1*	~40 hrs	No wireless connectors	Displays very well, all registers & memory are dynamic
CPU Sim	1**	~20 hrs	None	Does not actually have a data path and control circuits

Notes:
** The complete circuit is on one page, but includes four custom sub-circuits*
*** CPU Sim has a single run page, but many other pages are used to define the CPU*

We found each of these tools to be valuable for learning, but the synergy of having teams working on each and comparing and discussing was even more valuable. However if we were to have a single tool for a Computer Architecture class, the clear choice is Logisim.

REFERENCES

1. Patterson, D., Hennessy, J., *Computer Organization and Design, the Hardware/Software Interface*, San Francisco, CA: Morgan Kaufmann, 2005.

2. Null, L., Lobur, J., *The Essentials of Computer Organization and Architecture*, Sudbury, MA: Jones and Bartlett, 2004.

3. Lewellyn, M., Sprague, B., CEDAR Logic Simulator, http://sourceforge.net/projects/cedarlogic/, accessed 19 Jan 2011.

4. Burch, C., Logisim, http://sourceforge.net/projects/circuit/, accessed 19 Jan 2011.

5. Skrien, D., CPU Sim: An Interactive Java-based CPU Simulator for use in Introductory Computer Organization Classes, http://www.cs.colby.edu/djskrien/CPUSim/, accessed 19 Jan 2011.

6. Mills, G., Multimedia Logic, www.softronix.com, accessed 19 Jan 2011.

7. Stanley, Prigmore, Mikolyski, Embrey, Fife, (2007 June) "Pedagogic Value in Understanding Computer Architecture of Implementing the Marie Computer from Null and Lobur in the Logic Emulation Software, Multimedia Logic", Proceedings of the Workshop on Computer Architecture Education, 34th International Symposium on Computer Architecture, p 66-71.

INTEGRATING MICROCONTROLLERS IN UNDERGRADUATE CURRICULUM*

William Albrecht, McNeese State University, (337) 475-5816, walbrecht@mcneese.edu
Paul Bender, McNeese State University, (337) 475-5800, pbender@mcneese.edu
Kay Kussmann, McNeese State University, (337) 475-5801, kkussman@mcneese.edu

ABSTRACT

Current computer science programs fall short in introducing the use of electronics for multidisciplinary applications. This paper will present how one institution plans to incorporate throughout its computer science curriculum the use of a microcontroller for various multidisciplinary embedded applications. The primary goal is to engage students regularly with real world data acquisition and processing. The goal of this project will result in students realizing a wider scope of applications of computer science outside of traditional coursework.

INTRODUCTION

The computer science curriculum at McNeese State University (MSU) includes sixteen three-credit computer science courses. These courses account for forty-eight credits of the one hundred twenty required credits for the BS degree. At present no required computer science courses include actual programming of microcontrollers or embedded devices.

The southwest Louisiana and southeast Texas regions are home to both large petrochemical and related industries and huge pristine wetlands. Research projects in the area frequently require collection (often remotely) of vast amounts of data from various types of probes in inhospitable environments. Equipping graduates in our computer science program with the experience of interacting with embedded systems and microcontrollers will encourage them to apply their computer science background to real world problems.

With the advent of reasonably priced, flexible microcontroller environments such as Arduino and LeafLabs Maple it becomes feasible to incorporate these devices in a variety of undergraduate computer science classes. Incorporation of microcontrollers allows the student to make "computers that can sense and control more of the physical world than your desktop computer" [1]. The terms Cyber-Physical Systems and Interactive Computing are often associated with microcontroller based systems that sense and interact with the physical world [7, 9].

The four key components of our plan include: a classroom set of Arduino boards included in a SparkFun Inventor's Kit for Arduino (approximately $99.00 each), a sequence of computer science courses in which the applications of the Arduino environment can be explored, faculty trained on the fundamentals of the Arduino Platform, and students who thrive on hands-on active learning. An Arduino device is shown in Figure 1 [1].

Figure 1: The Arduino Uno

GENESIS OF THE IDEA

Four events have led the MSU computer science faculty to embark upon the project of equipping the graduates of the program with skills gained from interacting with microcontroller devices:

(1) Several students have chosen a senior capstone project involving Cyber-Physical systems. These projects have included Arduino based autonomous model vehicles and controlling home electronics.

(2) Computer science faculty are participating in research activities conducted at the Louisiana Environmental Research Center (LERC). This research utilizes Arduino based environmental sensor networks to facilitate research of faculty in the agriculture, biology, engineering, and environmental science departments.

(3) The computer science department is attempting to increase enrollment by attracting students looking for more physical hands-on application in their major.

(4) MSU's computer science department desires to set itself apart from other computer science departments in the state of Louisiana. The State of Louisiana has instituted significant cuts to University budgets. As a consequence of these cuts, programs that are considered to be duplicate programs at various institutions have to defend their unique qualities in order to avoid being considered for elimination.

VISION

Our vision is to introduce a simple hardware platform, Arduino, across the computer science curriculum which allows students to gain hardware interfacing experience by writing software that directly interacts with hardware devices.

The following objectives will be addressed throughout the project:

(1) Show students how to program microcontrollers and embedded devices.

(2) Show students how to interface with hardware.

(3) Build a recruiting and retention tool by bringing hands-on real world applications into computer science courses.

(4) Allow students to transition to jobs in local industry where hardware interaction may be required such as automated plant processes which use embedded systems.

Arduino hardware will be utilized in a variety of courses throughout the curriculum ranging from the Introductory CS I class to the senior capstone project course.

Lower Level Computer Science Course Integration

The lower level computer science students would have their initial interaction with the Arduino development system in three courses: CS I, CS II, and Numerical Methods I.

CS I is an introduction to programming using C programming language. Students would have their first interaction with the Arduino platform addressing the problem of simple input and output. An excellent reference for a gentle introduction to the Arduino platform is Getting Started with Arduino by Massimo Banzi [2]. Sample assignments would include:

(1) Controlling the on/off state and blink speed of a simple LED.

(2) Reading input from various types of digital input devices.

CS II is the second course using C programming covering topics such as pointers and functions. Continuing with examples such as those found in Banzi's book students would:

(1) Program sensors and actuators including analog input and output.

(2) Serial communication between a computer and the Arduino.

In the first numerical methods course, students will use Arduino hardware for data acquisition to interpret real time data and analyze by interpolation, line fitting, numerical differentiation, and numerical integration. Sample assignments would include:

(1) Writing a program to acquire real time data using a sensor connected to the Arduino.

(2) Writing a program to interpret real time data acquired by Arduino hardware using appropriate algorithms for interpolating, line fitting, numerical differentiating, and numerical integrating.

Upper Level Computer Science Course Integration

The upper level computer science students will continue their interaction with the Arduino development system in four courses: Operating Systems, Architecture, Networking, and Numerical Methods II.

In the senior level operating systems course, students will be asked to develop a basic operating system in phases using the Arduino platform. Some phases that will be addressed are:

(1) Write a device driver so that the Arduino may read and write data to a MicroSD card.

(2) Write a memory management routine, where the Arduino writes information to more than one addressable memory module.

In the senior level architecture class the Arduino will be utilized in the following activities:

(1) Use the Arduino design as a case study in computer architecture.

(2) Use the Arduino to interface with external logic circuits, constructed on a breadboard.

(3) Use the Arduino to interface with several devices and/or external memory using different I/O bus architectures. Shields that contain more than one device in an appropriate configuration may be used, or circuits may be constructed on a breadboard.

In the second numerical methods course, students will use Arduino hardware for data acquisition to interpret real time data and analyze data through approximation. Sample assignments would include:

(1) Write a program to acquire real time data using a sensor connected to the Arduino.

(2) Write an approximation algorithm to interpret real time data acquired using Arduino hardware.

In the senior level networking class the Arduino will be used to implement the following activities:

(1) Utilizing two Arduinos, create a simple computer network with the Arduino devices talking to each other over as few as two wires, using RS232 serial or an I2C communication bus.

(2) Utilizing an Ethernet Shield attached to an Arduino, construct a client or server for a wide range of protocols to talk to the Internet.

In addition to the prior courses listed above the curriculum includes a topics course, used by senior students to fulfill computer science elective credits. This course has significant potential for advanced Arduino application. Some example topics which will be offered utilizing the Arduino platform are:

(1) A course in robotics that can include activities centered on a mobile robotics chassis, such as the Ardubot from Sparkfun Electronics. This course explores the challenges involved in interfacing with control hardware and simultaneously responding to inputs from sensing devices.

(2) A course on sensor networks utilizing the Arduino, a wireless network shield, and environmental sensors which will explore the challenges of bringing data to a central location using a low bandwidth network, and meeting network constraints.

(3) A course in interactive computing which can be developed utilizing Arduinos to respond to human actions.

HARDWARE SELECTION

The Arduino platform is simple enough to describe in sufficient detail to complete a project in one or two class periods. This alleviates the concern of how to interleave the hardware into existing courses due to time constraints. Features of the Arduino platform which make it ideal for integration into our program are:

(1) The hardware design is an open-source prototyping platform, which means that complete reference materials for hardware and software are available for classroom work [1].

(2) The devices are programmed using a derivative of the C programming language, the language used as the basis of our CS I and CS II courses [1].

(3) The devices are cross-platform (Mac, Linux, PC) and may be connected to any computer by a USB interface, a common standard interface for all modern PC's [1].

(4) Arduinos are easily reconfigurable using stackable interface cards called "Shields." Shields allow tailoring the hardware environment to specific experimental configurations without requiring students to obtain a significant knowledge of hardware construction techniques. Figure 2 shows an Ethernet Shield, which includes an Ethernet Network port and a MicroSD card slot [1].

(5) An additional advantage to the Arduino programming environment is a secondary programming environment called ModKit. ModKit provides a drag-and-drop programming language to program the same hardware which would be ideal for transfer students who may not have C programming language or for demonstrations contributing to our recruiting efforts [1].

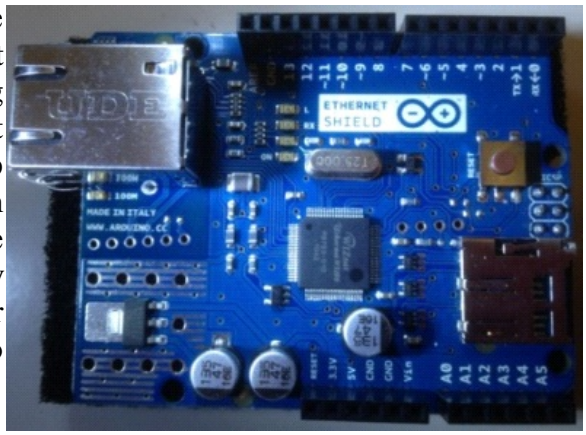


Figure 2: The Arduino Ethernet Shield

Various packages from other manufacturers are available, however our main concern is in providing a kit based on an open source hardware/software platform that would address all course needs at a reasonable cost. The Sparkfun Inventor's Kit for Arduino seems to fit that profile.

IMPACT ON CURRICULUM AND INSTRUCTION

An important component of implementing this project is training for faculty. We are fortunate to have one of our faculty members who has laboratory research experience in this field. This professor will be responsible for training faculty members and help with the design and setup of activities appropriate for each course. Keeping track of the inventory and upkeep of Arduino components will be another responsibility for the faculty member.

By introducing a new hands-on component to courses early in the curriculum, we hope to show students that computer science is more than just sitting at a desk programming computers. We expect this to ultimately attract and retain a larger number of students in our computer science department.

The Arduino platform will bring a unique flavor of instruction used by McNeese faculty and provide a new research direction for our department. Another advantage to the Arduino programming environment would be to provide an outreach program for local schools which will aid in attracting new students to our program.

CONCLUSION

As a result of this project Arduino hardware will be incorporated in the undergraduate computer science curriculum to give students more experience with hardware interaction. The following chart provides a summary list of potential uses of the Arduino hardware in some of our computer science curriculum.

Title	Potential uses
Introduction to Computer Science I	Basic input and output using C
Introduction to Computer Science II	Actuators and sensors using analog and digital input/output, serial communication with PC
Numerical Methods I	Data collected using an Arduino can be utilized as input to algorithms for interpolation, line fitting, numeric differentiation, and numeric integration
Numerical Methods II	Data collected using an Arduino can be utilized as input for approximation algorithms
Introduction to Operating Systems	Arduinos can be used to implement operating system features such as process scheduling, memory management, and file systems
Computer Organization and Architecture	Arduinos can be used as a computer reference design and to implement interconnection buses (e.g. I2C and SPI)
Introduction to Computer Networking	Networks of Arduinos or Networks consisting of Arduinos and PCs can be constructed
Topics	Topic dependent (eg. Robotics, sensor networks, etc.)

REFERENCES

[1] Arduino - Home Page, 2011, http://arduino.cc, 2011.

[2] Banzi, M., "*Getting Started with Arduino*," O'Reilly Media Inc., Sebastopol, 2009.

[3] Faludi, R. "*Building Wireless Sensor Networks*," Sebastopol, O'Reilly Media, Inc., 2011.

[4] Jamieson, P. "Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?," 2010, www.users.muohio.edu/jamiespa/html_papers/fecs_11.pdf, 2011.

[5] Margolis, M. "Arduino Cookbook," Sebastopol, O'Reilly Media, Inc., 2011.

[6] The National Science Foundation, "*Cyber-Physical Systems (CPS)*", 2011, http://www.nsf.gov/pubs/2011/nsf11516/nsf11516.htm, 2011.

[7] Noble, J. "*Programming Interactivity A Designer's Guide to Processing, Arduino, and openFrameworks*," O'Reilly Media, Inc., 2009.

[8] Oxer, J., Blemings, H. "*Practical Arduino Cool Projects for Open Source Hardware,*" New York, Springer-Verlag, 2009.

[9] Schmidt, M. "*Arduino A Quick-Start Guide*," Raleigh, Pragmatic Programmers, LLC., 2011.

[10] Waddington, N., Taylor, R. "*Arduino & Open Source Design*," 2007, http://sfu.academia.edu/NathanWaddington/Papers/292423/Arduino_and_Open_Source_Design, 2011.

SIMPLE COMPUTER SECURITY EXERCISES FOR EVERY CLASS ROOM*

CONFERENCE TUTORIAL

Nadine Hanebutte, Ph.D.
Department of Mathematical and Computing Sciences
St. John Fisher College
nhanebutte@sjfc.edu

Computer Security has grown to a field of significant importance over the last 25 years. Hacking and Cracking are subjects that many students are fascinated by and curious about. In addition, the national trend of Computer Science programs adding courses on Computer Security to their offerings indicates that students should have some general knowledge about what issues arise in the context of Computer Security.

During this tutorial a number of security related topics are demonstrated in order to provide educators with the knowledge about how to integrate these into general Computer Science courses in order to augment the course and allow students to appreciate the connection between certain security issues and conventional Computer Science topics. The demonstrated topics encompass among others: doorknob-rattling, spoofing, and information mining.

In addition to the demonstration it will be discussed what conventional topics are touched by each security demonstration and how those demonstrations could be incorporated into courses, for example, Networking, Operating Systems, Programming, and Computer Ethics.

Audience members who bring their own laptop will have the opportunity to participate in some of the demonstrations. The tutorial is geared towards Computer Science educators. A prior in-depth knowledge in the field of Computer Security is not expected.

AN EFFECTIVE EDUCATIONAL MODULE FOR BOOTH'S MULTIPLICATION ALGORITHM*

Christopher M. Jenkins, Computer Science Department, Trinity University, San Antonio, TX 78212, cjenkin1@trinity.edu

Adam D. Voss, Computer Science Department, Luther College, Decorah, IA 52101, vossad01@luther.edu

David Furcy, Computer Science Department, University of Wisconsin Oshkosh, Oshkosh, WI 54901, furcyd@uwosh.edu

ABSTRACT

We have developed a free online module for the self-study of Booth's multiplication algorithm. This module includes an algorithm visualization tool that displays both the pseudo-code and the contents of hardware registers during execution. This tool accepts user-generated inputs, randomly generates questions to foster active learning and comes with a user guide and a standalone hyper-textbook chapter that explains the algorithm, justifies it mathematically and includes exercises that are integrated with our visualization tool. Student grades and feedback suggest that this module is an effective platform for self-study.

1 INTRODUCTION

Since "[a] professional in any field of computing should not regard the computer as just a black box that executes programs by magic[, a]ll students of computing should acquire some understanding and appreciation of a computer system's functional components" [1]. Therefore, virtually all computer science curricula include at least one course on computer architecture, covering a wide range of topics from data representation to multiprocessing. Given the increasing importance of recent architectural trends, such as multiprocessing and distributed architectures, instructors end up having to make tough decisions about topic coverage. To make room for these high-level architectural trends,

---

there is a tendency to drop topics close to the hardware-software interface, especially at the logic design level [1]. In this paper, we consider one such lower-level topic within the first core area in the latest ACM/IEEE computer science curriculum recommendation, namely "signed and unsigned arithmetic" [1]. Given the wide variety of data representation schemes, algorithms and hardware implementations, some important ideas in this area are being covered less and less frequently. In this paper, we focus on one such topic, namely signed integer multiplication, for which Andrew Booth presented a seminal solution [3]. While his algorithm is still being covered in some modern textbooks (e.g., [10]), it is not even mentioned in others ([e.g., [11]). Another classic textbook by Patterson and Hennessy [9] best illustrates the recent fading of this topic. Their second edition devotes five full pages to it (pages 259-263) in the main body of the text. Their third edition devotes the same number of pages but moves them out of the main text and onto the accompanying CD. Their fourth edition does not cover this algorithm at all: in fact, the full-text search engine provided with the book yielded "no results found" over the entire text and CD contents when fed the single keyword "Booth."

Given the decreasing availability of textbook materials and class time for such low-level but nevertheless important topics, our goal is to provide support for instructors who would still like their students to gain familiarity with Booth's algorithm. Therefore, we have designed a self-contained educational module that includes an online textbook or hyper-textbook chapter and an engaging visualization tool. Our hypothesis is that our free, online learning module will enable students to study and master this topic with little or no class time devoted to it, and no textbook reference. In this paper, we describe this module and our preliminary test of this hypothesis. In the next section, we describe related work. Then we describe our hyper-textbook and visualization tool in the next two sections, respectively. We then report some preliminary results on the pedagogical effectiveness of our module. We conclude this paper with a discussion of future work.

2 RELATED WORK

Booth's multiplication algorithm is still covered or at least mentioned in several textbooks [e.g., 5,8,10]. The most common approach to teaching Booth's algorithm uses pseudo-code or a flowchart as well as one or more traces of it on canned inputs. Some presentations justify mathematically why the algorithm works for both positive and negative numbers. Some include a description of "Booth recoding," which is a way to rewrite the multiplier to reformulate the multiplication as a sequence of additions and subtractions. While a discussion of Booth recoding is not strictly necessary to explain the algorithm, our module not only discusses it, but also includes pseudo-code and a mathematical justification for the algorithm's correctness and performance characteristics. The next two sections provide more details on the algorithm.

The main shortcoming of standard textbooks is their static nature. Since they only contain canned contents, they do not naturally support hands-on, interactive learning. Once the student has read the pseudo-code and included example(s), textbooks do not have any affordances for additional practice or active self-study. In contrast, our module enables students to trace through an arbitrarily large number of examples with randomized or user-selected input values and register sizes. The tracing of the algorithm is step-by-step, interactive, with the possibility to move both forward and backward through the trace. Randomized stop-and-think questions pop up to promote active

learning. The use of algorithm visualization also aims to foster learning in visual thinkers. The only other publicly available visualization of Booth's algorithm we found is a JavaScript simulator that, like our module, allows the learner to pick the two numbers to multiply, in either binary or decimal format [5]. However, this simulator is not interactive after the data entry stage. In fact, it produces the whole trace in one shot (no animation). Unlike our module, this simulator does not include pop-up questions or exercises. On the other hand, the web site associated with the textbook (see [5]) includes simulations for other arithmetic algorithms, which our module does not yet.

The Algorithm Visualization Portal [2] does not include any visualization of Booth's algorithm. However, our literature search yielded two relevant papers. Garzón et al. describe an approach to teaching computer architecture that includes "carefully designed practical exercises […] supported by an auxiliary computer-based environment" [4], which, to the best of our knowledge, are not publicly available. Finally, Thiebaut describes an architecture course in which students must build animations of algorithms, including Booth's [12]. His practice is the opposite of ours since class time is devoted to GUI programming and teaching the API to Trolltech's Qt GUI software toolkit before getting to the architecture topics proper. While the activity of building (as opposed to using) algorithm visualizations promotes a higher level of student engagement [7], the class time devoted to non-architectural topics makes it, in our opinion, less viable, given the recent trends described earlier.

3 HYPER-TEXTBOOK FOR BOOTH'S MULTIPLICATION ALGORITHM

The hyper-textbook chapter in our pedagogical module contains nine sections. The first section describes the prerequisite knowledge that is expected of the reader, namely mastery of the two's complement representation of signed integers and the basic operation of addition. The rest of the chapter is a motivated and progressive unveiling of Booth's algorithm that builds on students' existing knowledge. The second section explains why multiplying two numbers p and q via q additions of p is not an acceptable hardware implementation approach since its runtime is unpredictable. The third section considers an alternative approach to multiplication, namely the shift-and-add or paper-and-pencil approach, and introduces important terminology, such as multiplicand, multiplier, partial product, etc. The fourth section describes two improvements to this algorithm that are motivated by hardware constraints: the use of a running total of the partial products, since hardware typically handles the addition of only two numbers, and the switch to a sign-preserving or arithmetic right shift, which more efficiently lines up the two numbers to be added and sets aside one more bit of the final result as soon as it has acquired its correct value. The fifth section illustrates with an example why this algorithm does not handle negative numbers correctly. The sixth section explains Booth's recoding of the multiplier in terms of additions and subtractions and why this transformation corrects the mishandling of negative multipliers. At this point, students have been given a complete justification for the specification of Booth's algorithm, whose pseudo-code is shown on the right side of Figure 2 below. The seventh section brings the description of the algorithm down to the level of hardware registers. The eighth section contains a hyperlink to our visualization tool. The ninth and final section is comprised of three types of exercises. In some exercises, our visualization tool randomly generates numbers to multiply and asks students to fill in initial, intermediate or final register values, which

they can only do by tracing the algorithm. In contrast, the last kind of exercises ask students to input two numbers whose multiplication will exhibit the worst run time of Booth's algorithm. These exercises target a deeper level of understanding, beyond the tracing of the algorithm. Indeed, students are responsible for inferring that replacing a series of additions corresponding to a series of consecutive ones in the multiplier only speeds up the algorithm when the series of ones is long enough. What is common to all exercises is that the size of the registers is randomly generated, as are all of the initial numbers provided to students. More importantly, the exercises are fully integrated with our visualization tool: after students have entered their numerical answers, our visualization tool traces the algorithm based on them and identifies any errors students may have made at the time when these errors come up in the trace.

4 VISUALIZATION TOOL FOR BOOTH'S MULTIPLICATION ALGORITHM

Since our visualization tool uses the JHAVE platform, we start by highlighting the architectural and user interface features of JHAVE that are relevant to this project.

4.1 The JHAVE Educational Platform

The Java-Hosted Algorithm Visualization Environment (JHAVE [6]) supports a variety of algorithm visualization (AV) engines by providing them with:

- a common drawing context called the "main pane,"
- a standard set of DVR-like controls that allow students to step through a series a snapshots, one for each step in the algorithm being visualized in the main pane,
- two side panes where dynamically-generated HTML pages display accompanying text, such as pseudo-code and explanations of the visualization,
- input generators that gather student-generated input data needed by the algorithm,
- stop-and-think questions that pop up randomly during the visualization to foster active learning, and
- a database that keeps a record of all student answers for automatic assessment purposes.

JHAVE has a script-based, client-server architecture. In a typical session, the student web-starts the client program and selects an algorithm to visualize. If input is required, the server sends back an input generator. Once the student has entered the input data, the server runs the algorithm and generates a script, that is, a description of snapshots that portray the behavior of the algorithm. The AV engine (client) then retrieves this script and visualizes it in the main pane, moving forward and backward through the snapshots as the student clicks on the DVR-like controls, asking questions at random times and recording the student's answers. Therefore, the JHAVE platform supports a high level of interactivity. Besides zooming and panning, students can navigate through the visualization both forward and backward to help them identify as quickly as possible at what step of the algorithm their mental model breaks down.

4.2 User Interface

Our module includes a detailed user guide to our visualization that we summarize in this section. First, students point their favorite web browser to http://jhave.org and web start the JHAVE client (a Java application that will run on any platform) with a single click. In the client's window, students enter the name "booth" in the category field to select one among dozens of algorithms available. After clicking "Connect" (to the server) and "Visualize" (the algorithm), students are presented with an input generator (see Figure 1) with four text fields that enable them to pick the values of two numbers they want to multiply, and one drop-down menu to select the size of the registers in bits.

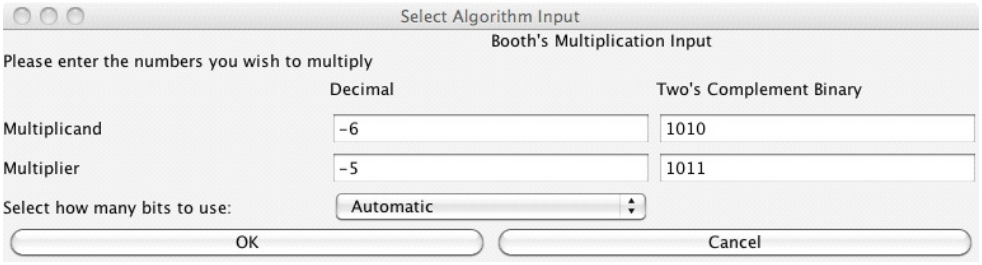


Figure 1: Input generator window

To make this tool as user-friendly as possible, the input generator automatically converts each input number from decimal to two's complement binary and vice-versa. Similarly, the default value for the register size is computed automatically but can be overridden with the drop-down menu. Finally, an error message is displayed if one of the numbers overflows the selected register size. After students have completed the input stage, they click the "OK" button to bring up our main visualization window. After going through the first few steps of the algorithm, the window will look like Figure 2, with the pseudo-code and graphical depictions of the algorithm in the right and left panes, respectively. Each pseudo-code line is numbered and highlighted in sync with the graphical depiction. The top corners of the left pane always display the multiplication being visualized in both decimal (left corner) and binary (right corner) formats, with a textual description of the current step between these two multiplications. The trace of the algorithm appears below this top line, with the first row of boxes depicting the initialization steps on pseudo-code lines 2 through 6, as shown in Figure 2. Every row under this initialization row depicts the execution of one iteration of the main loop on lines 10 through 21. Figure 3 depicts the final contents of all registers with the result of the multiplication in registers A and Q. Lack of space prevents us from including intermediate snapshots in which the Math/ALU column contains a detailed depiction of the subtraction or addition that is performed on lines 11 or 14, respectively. Finally, our visualization tool randomly generates contextually appropriate questions that pop up to engage the learner in active learning instead of passive viewing. Our visualization leverages all four types of questions supported by JHAVE, namely true/false, multiple choice, short answer (see Figure 4) and multiple selection (see Figure 5) questions.

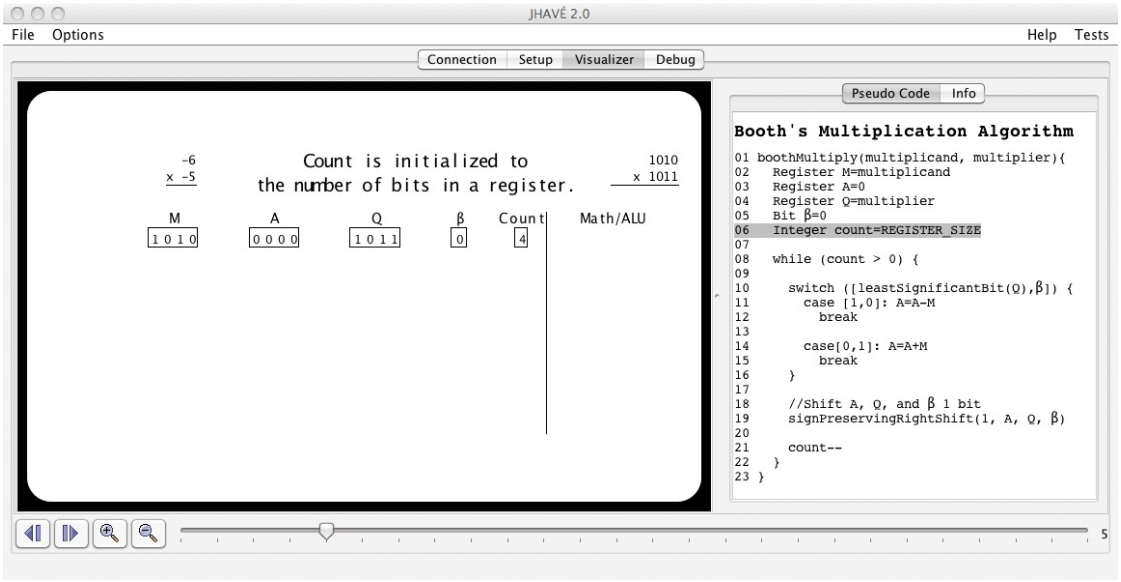


Figure 2: Algorithm visualization during execution

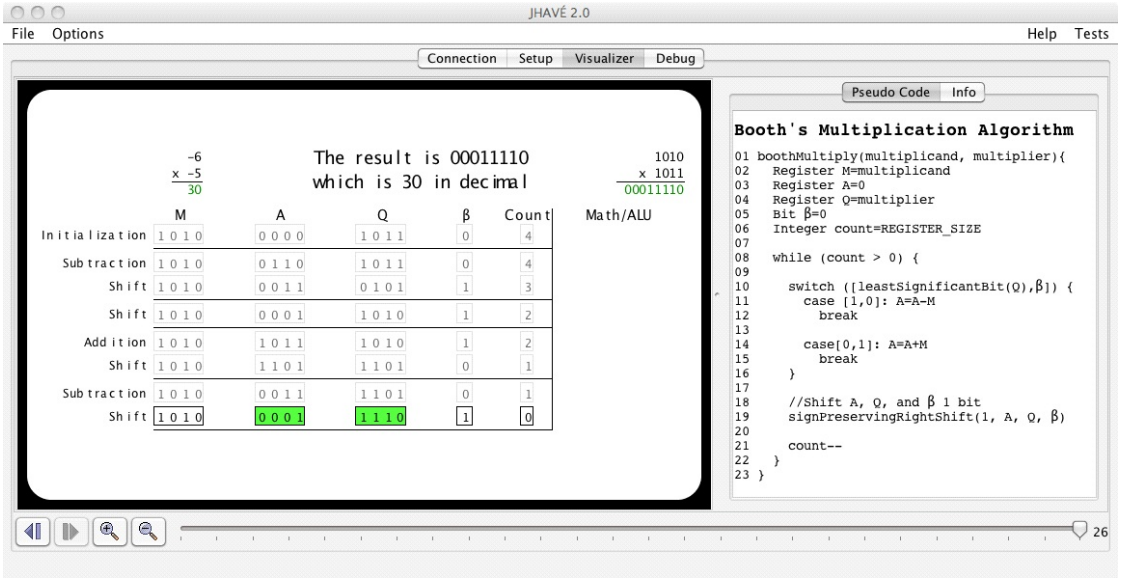


Figure 3: Algorithm visualization after termination

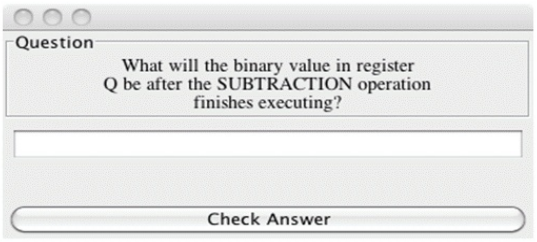


Figure 4: Pop-up question

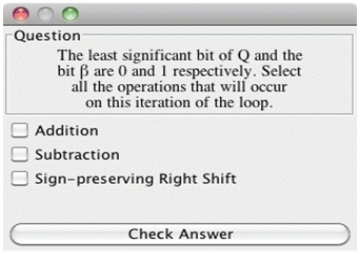


Figure 5: Another pop-up question

5 PEDAGOGICAL EFFECTIVENESS: PRELIMINARY RESULTS

In the Fall 2011 semester, the third author used our module in a junior/senior-level course on computer organization that is a follow-up to a sophomore-level course on computer architecture and assembly language. After a lecture and a ten-minute quiz on the pencil-and-paper (P) algorithm for multiplication, students were given one week to peruse our module on their own (no lecture), after which they took two consecutive 10-minute quizzes. The first quiz (B1) required students to trace the Booth algorithm on a randomly generated pair of numbers and a fixed register size. The second quiz (B2),

	P	**B1**	**B2**
Minimum score / #students	6 / 1	6 / 3	6 / 3
Maximum score / #students	10 / 4	10 / 8	10 / 6
Average score	8.44	8.50	8.39
(Standard deviation)	(1.21)	(1.54)	(1.64)

Table 1: Quiz results

taken right after B1, first asked students to compare the run time performance of both the paper-and-pencil and Booth's algorithms in terms of the number of primitive operations executed when run on three different input pairs. Then, the second half of quiz B2 asked students to produce the only input numbers that would exhibit the best and worst performances by Booth's algorithm. Table 1 reports the minimum, maximum and average scores for all three quizzes (sample size of 18) out of a maximum possible score of 10. The first observation is that the average quiz scores are rather close, even though quiz P was taken after a lecture on the topic, while the preparation for quizzes B1 and B2 was through self-study exclusively. However, both the variance and number of students who received the minimum and maximum scores are larger for both quizzes on Booth's algorithm. We conjecture that self-study, in this case, accentuated individual differences in motivation, perseverance and time on task. We also observe that more students earned a maximum score on B1 and B2 than on quiz P. This is surprising to us, since quiz B2 assessed a much deeper understanding of the algorithm, namely prediction of overall performance without tracing, compared to quizzes P and B1, both of which required mechanical tracing of the execution. We believe that a significant number of our students

mastered the algorithm on their own. In a 13-question survey administered anonymously after quiz B2, students reported a wide range of study times (from no time at all to 3 hours) and frequency of use of our visualization tool (from 0 to 12 times or up to 30 minutes). When asked how helpful they found our tool, students gave an average score of 4.35 on a scale from 1 (useless) to 5 (indispensable). Finally, 6 out of 18 students reported having used the Wikipedia entry for Booth's algorithm during their self-study, which, as retrieved on 11/2/2011, does not discuss the performance aspects assessed in quiz B2.

6 FUTURE WORK AND CONCLUSION

Following up on the feature requests obtained through our survey, we are working on improving the level of detail in the feedback given to students when they give a wrong answer to a pop-up question. In addition, we plan to conduct a thorough empirical study to characterize and improve the pedagogical efficacy of our module. Finally, we would like our module to cover a variety of additional arithmetic algorithms (e.g., division) that are well within the scope of our visualization tool and hyper-textbook.

7 REFERENCES

1. ACM/IEEE Interim Review Task Force, Computer Science Curriculum 2008: An Interim Revision of CS 2001, http://www.acm.org/education/curricula-recommendations, 2008
2. The algorithm visualization portal, http://algoviz.org/.
3. Booth, A., A signed binary multiplication technique, *The Quarterly Journal of Mechanics and Applied Mathematics*, IV, (2), 236-240, 1951.
4. Garzón, E., García, I., Fernández, J., An approach to teaching computer arithmetic, *International Meeting on High Performance Computing for Computational Science*, 2002.
5. Koren, I, *Computer Arithmetic Algorithms*, 2nd Edition, A. K. Peters, 2002.
6. Naps, T., Rößling, G., JHAVE -- More visualizers (and visualizations) needed, *Electronic Notes in Theoretical Computer Science (ENTCS)*, 178, 33-41, 2007.
7. Naps, G., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velázquez-Iturbide, J., Exploring the role of visualization and engagement in computer science education, *ACM SIGCSE Bulletin*, 35, (2), 2003.
8. Null, L., Lobur, J., *The Essentials of Computer Organization and Architecture*, 3rd Edition, Jones & Bartlett Learning, 2012.
9. Patterson, D., Hennessy, J., *Computer Organization and Design*, 2nd 3rd and 4th Editions, Morgan Kaufmann, 1998, 2005, and 2009.
10. Stallings, W., *Computer Organization and Architecture*, 8th Edition, Prentice Hall, 2010.

11. Tanenbaum, A, *Structured Computer Organization*, 5th Edition, Prentice Hall, 2006.

12. Thiebaut, D., On startups and teaching computer architecture, *Journal of Computing Sciences in Colleges*, 22, (6), 28-36, 2007.

8 ACKNOWLEDGEMENTS

This research was funded by an NSF REU grant (award #0851569).

BUILDING A BETTER UNIVERSITY iPHONE APPLICATION*

Ongard Sirisaengtaksin, Ph. D., Maxwell Goedjen, and Brian Holtkamp
Department of Computer and Mathematical Sciences
University of Houston-Downtown, Houston, TX 77002
713 221-8554
sirisaengtaksino@uhd.edu

ABSTRACT

The first day of classes for a new student at a college or university can be very confusing. The first thing a student usually tries to do is look up a class schedule and then try to locate his or her classroom. Since many students these days have a smart phone, it would be very convenient and helpful if there was a mobile application capable of providing step-by-step directions for students to find a classroom and access their class schedule from their mobile phones. Therefore, the main objective of this project is to build an iPhone application to assist new students at a college or university to locate a classroom in the campus by displaying the path from where they currently are to where their destination is. In case of emergency situations, the app is also designed to display a path to the nearest exit. Moreover, the app is set up to view their class schedule as well. To be able to construct step-by-step directions to assist students, a map must be built specifically for the app. Building the map is a three step process. First, the visual component of the map is created in Google SketchUp from floor plans. Next, all the rooms of each map or floor plan are annotated and marked down for possible paths. Then, XML is used to annotate the maps. Once the app has this information, it is capable of dynamically routing students to and from any room in the school.

INTRODUCTION

The motivation for this project stemmed from the problem a new student at a college or university encounters in locating a classroom on the first day of classes. Since most students are not familiar with the campus map, it can be a very discouraging situation. These days, most students have a mobile phone, and it would be very convenient and helpful if there were a mobile application that can assist students to find a classroom and

---

access their class schedule from their mobile phones. The iPhone was chosen for this project because it is very popular among students. Furthermore, the iOS SDK [2] used to develop iOS or iPhone apps is very well established and very stable. To be able to construct step-by-step directions to assist students, a map must be built specifically for the app. Building the map is a three step process. First, the visual component of the map is created in Google SketchUp [3]. Next, the map is annotated, marking down rooms and possible paths around the school. Then, XML [4] is used to annotate the maps. Once the app has this information, it is capable of dynamically routing students to and from any room in the school. This app also includes the capability of notifying students of emergency situations. It is designed to provide notification for two types of emergencies, urgent emergencies and advisories. In both cases, an emergency contains a title, description, and an expiration date. In the case of an advisory, a blue bar is shown at the top of the application, and when tapped, it displays the information listed above. When an urgent emergency is in effect, the app is in a non-dismissible mode. The app will display the emergency information, as well as prompt the student for his/her current location. When the location is entered, the app will use its path-finding capabilities to generate and display a route to the nearest exit, specifically avoiding the use of elevators.

CONSTRUCTION OF MAPS IN GOOGLE SKETCHUP

The maps for the iPhone application can be constructed by first, outlining the main features such as the walls, stairs, and elevators in the floor plans of the college or university buildings. Then a 3 dimensional model is created using Google's 3D model editor, called Sketchup, of the floor in question. A color scheme is used to organize the rooms that are queried and their respective hallways. Once the 3D model is created, an aerial rendering of the map that is seen within the iPhone application can be generated using walls, stairs, and elevators from the 3D model. Then, a path-finding scheme can find the most efficient route from destination to destination.

MAP ANNOTATION

A program named "MapTagger" is designed to build and generate XML files to annotate the map. This application takes a rendered image of the Google SketchUp 3D Maps file and allows the tagger to add "nodes" to the map. Every node contains a set of x and y coordinates as well as a Unique IDentification (UID) for the node. All of these nodes are processed into an XML file which can be read by the path-finding process.

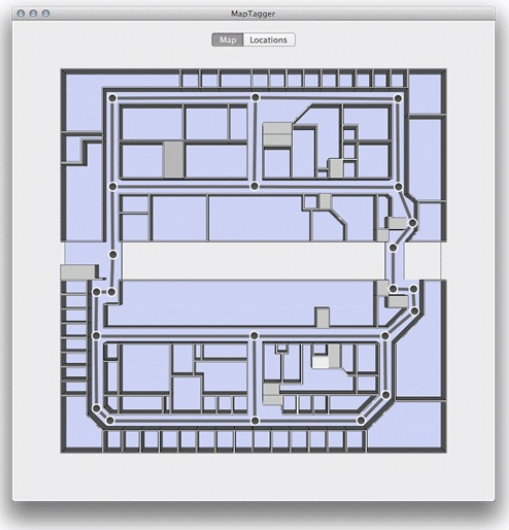


Figure 1: Annotating Control Points

First, the map is annotated with "control" nodes. These nodes define the possible paths that a user can take through the building. The tagger adds control nodes to corners and any other place where the

user can change direction, and then connects these points to form paths. Control nodes themselves are not possible destinations for the user, they are only used to connect the user to his or her destination. A control node also contains the UIDs of the other control nodes that it is connected to, see Figure 1. The following is an example of a node with its UID and neighbors' UIDs:

```
<node>
    <name></name>
    <x>743</x>
    <y>232</y>
    <uid>E055DB4E-19B9-416A-9F4A-7BA8197F9A9A</uid>
    <neighbors>
     <neighbor>31F7519B-4D88-4D4E-B9CE-7D9C030D4E77
         </neighbor>

         <neighbor>F5990039-01EF-4F2F-A087-A493B4C9DD6A
         </neighbor>

         <neighbor>69667B82-F5C0-4B37-9FCA-631B3D565D2C
         </neighbor>
    </neighbors>
</node>
```

Next, "destination" nodes are added. These can be categorized as rooms, bathrooms, vending machines, elevators, etc, and, if applicable, can be given a name. A fully annotated map is shown in Figure 2. A destination node will also contain the UIDs of the two closest control nodes. A typical destination node will look like the following:

```
<node>
    <name>S711</name>
    <x>767</x>
    <y>1287</y>
    <uid>6F3CFE7A-252B-4F14-8FCD-463841D5D426</uid>
    <adjacentNodes>

     <adjacent>9E4F2468-0053-4EEA-9E77-C0884A283EF5
     </adjacent>
            <adjacent>CF79B4A3-8C9C-4006-B083-CD8E8EF74C61
     </adjacent>
    </adjacentNodes>
</node>
```

PATH-FINDING

Once the node information is loaded from the XML file, a modified version of the A* path-finding algorithm [1] is used to find the route. The app prompts the user for a starting point and a destination. Due to the lack of precision of GPS in a large building, the user is prompted to enter the room number closest to them. In a larger campus, the app might use GPS in order to get a coarse location for the student and prompt the student for a more precise location. The app looks up the nodes for the starting point and ending

point and stores them. The app then generates a "virtual" node between the two nearest control points, such that the intersection of the line formed between the control points and the line formed by the start and the virtual node is perpendicular. This virtual node is represented internally using a control node, and is given the start's adjacent nodes as its neighbors. The A* algorithm takes this node as the starting point, and traverses the neighbors of each successive node until it reaches one of nodes adjacent to the destination node. Another virtual node that connects the destination is generated the same way.

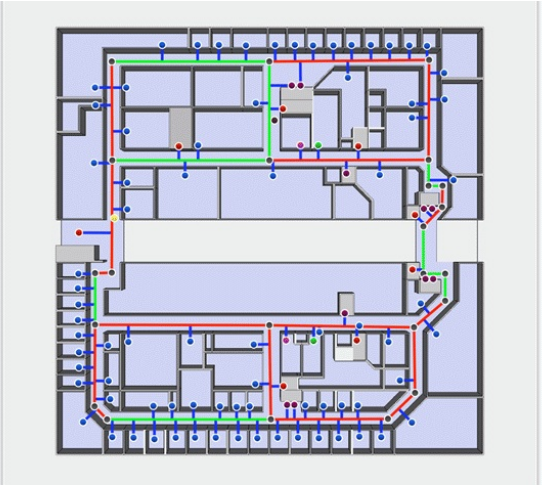


Figure 2: A Fully Annotated Map

Once the path has been calculated, it can be displayed as shown in Figure 3. First, the image of the relevant map is drawn onto the screen using a UIImageView [6], a class provided by the UIKit framework [7]. Next, the path is drawn using a UIBezierPath [5], a class provided by the UIKit framework [7] on top of the image using the coordinates provided in the nodes that were traversed.

EMERGENCY NOTIFICATION AND EVACUATION CAPABILITIES

At a college or university, a variety of different emergency conditions can arise that students need to be notified of. This app also includes the capability of notifying users of these emergencies. An XML file is maintained on a separate server which describes the condition of the emergency state of the university. There are two categories of emergencies, urgent emergencies and advisories. In both cases, an emergency contains a title, description, and an expiration date. In the case of an advisory, a blue bar appears at the top of the application as shown in Figure 4, and when tapped, it displays the emergency information listed above. When an urgent emergency is in effect (for a situation such as a fire), the app is "taken over." A non-dismissible modal popup is displayed showing the emergency information and prompts the user for their current location. When the location is entered, the app will use its path-finding capabilities to generate and display a route to the nearest exit, specifically avoiding the use of elevators.

Figure 3: Display of a Requested Path

SCHEDULE VIEWING

The app also allows the user to view their schedules as shown in Figure 4. It currently provides this capability by "scraping" the university website - reading in the standard output of the website and parsing it by looking for certain HTML elements. To "scrape" the website, the app needs to go through the same steps a user would in using the website to find their schedule. The app must authenticate against the university's website, download the HTML returned, and proceed to the page on the website where the student can view their schedule. The app downloads this page as a string, and uses regular expressions and string searches in order to "guess" where the applicable content is. In the future, we hope to have an official API to use for this feature, which is more reliable and easier to work with.

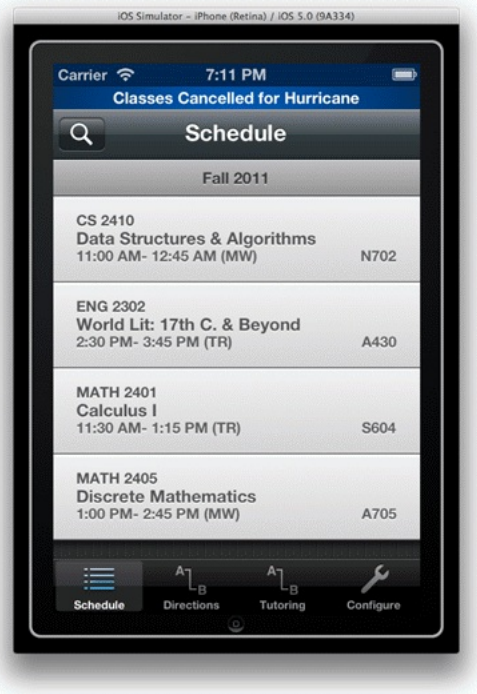


Figure 4: Schedule Page Displaying Advisory

CONCLUSION

The Path-Finding app is created using Google SketchUp 3D modeling tools to create maps of the floor plans. Then, a color scheme is used to organize it into the rooms that are queried and their respective hallways. Once the 3D model is created, the aerial rendering of the map that is seen within the iPhone application can be generated using walls, stairs, and elevators from the 3D model. The modified A* algorithm is implemented as a path-finding scheme to find the most efficient route from destination to destination. This app also includes the capability of notifying students of emergency situations. It is designed to provide notification for both urgent emergencies and advisories. In an urgent emergency case, the app will display the emergency information, as well as prompt the student for his/her current location. When the location is entered, the app will display a route to the nearest exit, specifically avoiding the use of elevators.

ACKNOWLEDGEMENT

This research is supported by NSF grant "Undergraduate/Graduate Student Immersion in Computer Science, Technology and Mathematics," DUE-0965952.

REFERENCES

1. Hart, P. E.; Nilsson, N. J.; Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on Systems Science and Cybernetics*, 4, (2), 100-107, 1968.

2. Apple iOS Developer Kit, http://developer.apple.com/devcenter/ios/index.action
3. Google SketchUp 3D Modeling Application, http://sketchup.google.com/, 2000.
4. World Wide Web Consortium, Extensible Markup Language (XML), www.wc3.org/XML/, 1998.
5. UIBezierPath Class Reference, http://developer.apple.com/library/ios/DOCUMENTATION/UIKit/Reference/UIBezierPath_class/UIBezierPath_class.pdf, 2010.
6. UIImageView Class Reference, http://developer.apple.com/library/ios/documentation/uikit/reference/UIImageView_Class/UIImageView_Class.pdf, 2010.
7. UIKit Framework Reference, http://developer.apple.com/library/ios/#documentation/uikit/reference/UIKit_Framework/_index.html, 2011.

THE SECURITY ASSESSMENT TEACHING CASE MODULE*

James W. McGuffee
St. Edward's University
Austin, TX
(512) 448-8465
jameswm@stedwards.edu

Jagadeesh Nandigam
Grand Valley State University
Allendale, MI
(616) 331-3639
nandigaj@gvsu.edu

B. Bhagyavati
Columbus State University
Columbus, GA
(706) 507-8170
bhagyavati@columbusstate.edu

Walter W. Schilling, Jr.
Milwaukee School of Engineering
Milwaukee, WI
(414) 277-7370
schilling@msoe.edu

ABSTRACT

Digital Home is a comprehensive software case study developed by software engineering faculty at Embry-Riddle Aeronautical University. This paper describes the Digital Home case study project, its application to computing education, and the development of a security assessment case module for the case study project. The Security Assessment case module was developed by a team of computing educators from various institutions during the "Teaching with a Software Life-Cycle Case Study" workshop held in June 2011.

USE OF CASE STUDIES IN SOFTWARE ENGINEERING EDUCATION

Over the past decade there have been various proposals to make changes in software engineering education in an attempt to bridge the gap between concepts taught in the academic setting and actual software engineering practice that occurs in industry [1, 2, 7, 8, 9]. One of the major concerns has been that students in the academic classroom have had limited or no exposure to the big picture of developing a complex software project and are unable to grasp the interrelatedness of all the parts involved in the software development process. The use of case studies has been suggested as an appropriate tool to assist in this matter. Specifically, Garg and Varma define a software engineering case as "an account of a software engineering (development) activity event

or problem containing background and complexities encountered by a software engineer" [2].

In addition to the usefulness of information and experience provided to the software engineering student, the case study is a useful pedagogical tool in that it encourages active learning. Students are not merely passive recipients of information. Rather, a good case study encourages participation, debate, and a more robust understanding of the material than is possible with the traditional lecture method in software engineering education [3]. Studies have also shown that when given a choice students prefer the case study approach instead of a more traditional lecture approach to software engineering education [2].

Another concern in software engineering education is that students should be exposed to software engineering concepts and ideas throughout the computer science curriculum and should not see software engineering as an isolated set of concepts taught in one or two courses. There is great promise in being able to use the case study approach to present software engineering concepts in a wide variety of computing courses. By having a resource of artifacts and tools a group of students could potentially use a case study to discuss the software engineering implications in almost any computing course they take [4, 6, 9].

In summary, the case study is seen as an effective tool in software engineering education for three main reasons:

1. case studies can help bridge the gap between theory and practice,
2. case studies are an active learning activity, and
3. case studies can be used throughout the computer science curriculum to increase students' exposure to software engineering principles.

DIGITAL HOME CASE STUDY PROJECT

Faculty and students at Embry-Riddle Aeronautical University developed the Digital Home case study project. The project began in 2006 as part of a National Science Foundation funded project named "The Network Community for Software Engineering Education" (NSF-0080502) [3]. More recently, work on this project has been funded through the National Science Foundation's "Curriculum-wide Software Development Case Study" (DUE-0941768) [6, 10].

There are four major objectives of the Digital Home case study project. The first objective of the project is to produce realistic software development artifacts to provide educators with teaching examples that have a "real world" feel. The second objective of the project is to organize these "real world" artifacts into a set of case modules (or mini case studies) which can be used throughout a computing curriculum. In other words, these case modules aren't to be used solely in the software engineering class. A third objective of the Digital Home case study project is to deliver software development resource materials, which can be adapted by individual educators to allow for variances in teaching and learning styles. This third objective is primarily accomplished through the Software Development Case Study Project website located at <softwarecasestudy.org>. The fourth major objective of the project is to actively engage computing educators in the use, assessment, and development of case study materials.

One of the ways in which this fourth objective was met was through a workshop held on June 13015, 2011 entitled "Teaching with a Software Life-Cycle Case Study Faculty Workshop." It was during this workshop that the authors of this paper met and developed the Security Assessment case module that is described in the next section [3, 5, 10].

Digital Home is a comprehensive case study project that can be used throughout a computing curriculum. There are two main parts of the case study. The first part is a robust set of realistic artifacts. As of June 2011, there were sixteen fully developed artifacts and nine additional artifacts in the process of being developed. The second part of the project is a set of teaching case modules. As of June 2011, there were five fully developed case modules and two in progress. During the workshop in June our team developed an additional case module for the Digital Home case study project.

THE SECURITY ASSESSMENT CASE MODULE

Below is the full text of the Security Assessment case module developed by our team as part of the "Teaching with a Software Life-Cycle Case Study Faculty Workshop."

Case Module:
Security Assessment

Prerequisite Knowledge:
One year of programming in a high-level language.

Learning Objectives:
Upon completion of this module, students will have increased their ability to:
1. Appreciate security issues inherent in hardware and software due to wireless transmissions.
2. Gain experience in working with real-world technology.
3. Understand the impact of security technology on architecture and design of the system.
4. Prioritize risks, identify vulnerabilities, and provide recommendations.

Keywords:
access control, authentication, trusted entities, risk analysis

Case Study Artifacts*:
DH High Level Requirements Definition
DH SRS ver1.4
DH Software Design Specification
DH Bios
*note: These artifacts are available online at <softwarecasestudy.org>

Case Study Participants:
Li Shen (System Architect – Team Leader)
Michel Jackson
Georgia Magee
Sumeera Nangia
Massood Zewail

Scenario:

In late August of 2010, Home Owner Inc. (the largest national retail chain serving the needs of home owners) established a new Digital Home Owner division that was set up to explore the opportunities for equipping and serving "smart houses" (dwellings that integrate smart technology into every aspect of home living). In August and September of 2010, the Marketing Division of Home Owner conducted a needs assessment for a Digital Home product that would provide the computer and communication infrastructure for managing and controlling the "smart" devices into a home to best meet the needs and desires of homeowners.

In early September 2010, a five person team was assembled for the project and started a "project launch". After project planning was completed the team began work on requirements analysis and specification. The first version, 1.0, was completed in early October 2010 and versions 1.1 and 1.2 were completed by late October.

Jose Ortiz's Facebook account has been recently hacked into. Specifically, inappropriate personal messages were sent to all the friends on his list. He has suddenly developed an intense interest in security. He has asked the Digital Home software team to prepare a security report for the Digital Home project.

Exercise:

1. As preparation for the case method, ask each student to read the case study artifacts listed above.

2. Divide the class into a set of teams with five students in each team. Each student takes on a role of one of the case study participants: Li Shen, Michel Jackson, Georgia Magee, Sumeera Nangia, or Massood Zewail.

3. Each team takes on the role of the Digital Home Team and prepares for a meeting with Jose Ortiz. The team should carry out the following activities:

a. Analyze the current SRS and identify potential security concerns in the current architecture.

b. Create a trust boundary diagram

c. Prepare a checklist with recommendation to mitigate security concerns.

d. Rework: The author makes changes according to checklist.

e. Follow-up: Changes are checked and verified against the diagram and checklist.

Appendices:

Exercise Booklet

Resource Information:

1. McGraw, Gary, Software Security: Building Security In, Addison-Wesley, 2006.
2. Maxim, Merritt, and Pollino, David, Wireless Security, McGraw-Hill, 2002.

Teaching Notes:

1. Class discussion and a reading assignment on trusted entities should proceed the exercise (e.g., from [McGraw] and [Maxim, Pollino]).
2. This case module could be used in different level courses and different types of courses. For example, it could be used in secure software development, security

architecture, computer networks, information assurance technologies, or database and system security classes.

3. Although this case module is designed as a team exercise to be completed in class, there are a couple of other ways the case could be used:
   a. This could be a teacher-led discussion of the issues related to designing secure software and vulnerabilities in the Digital Home project. The discussion could follow the tasks listed in the above exercise section.
   b. This case module could be assigned as a team assignment to be completed outside of class.
4. Assuming an adequate student preparation for the exercise, allowing student teams about 1-1.5 hours each for the exercise should be sufficient.
5. It would be beneficial to follow the exercise with a ten minute presentation from each team. Some key points to include in the presentation are the following:
   a. Discuss how the diagram was obtained.
   b. Discuss how well the checklist protects against the vulnerabilities.
   c. Students should be aware that instructor will randomly select a student from each team to present the team effort to the rest of the class.
6. A nice extension to this exercise would be to assign a follow-up take home exercise to modify the diagram and the checklist, assuming that Jose Ortiz wants the Digital Home Team to use VPN extensively.

CONCLUSION

This paper has discussed the motivations and reasons for using case studies in software engineering education. Specifically, the Digital Home case study project was used as a primary example for the approach to using case studies in software engineering education. Finally, the details of a Security Assessment teaching case module that was created by a team of computing educators as part of a National Science Foundation sponsored workshop was presented.

REFERENCES

1. Butler, S.A., A client/server case study for software engineering students, *12th Conference on Software Engineering Education and Training,* 156, 1999.
2. Garg, K. and Varma, V., A study of the effectiveness of case study approach in software engineering education," *20th Conference on Software Engineering Education & Training*, 309-316, 2007.
3. Hilburn, T.B., Towhidnejad, M., Nangia. S., and Shen, L., A case study project for software engineering education, *36th Annual Conference on Frontiers in Education*, 105, 2006.
4. Hilburn, T.B. and Towhidnejad, M., A case for software engineering, *20th Conference on Software Engineering Education & Training*, 107-114, 2007.
5. Hilburn, T.B., Massood Towhidnejad, M., and Salamah, S., The Digital Home case study material, *21st Conference on Software Engineering Education and Training*, 279-280, 2008.

6. Hilburn, T.B., Towhidnejad M., and Salamah, S., Read before you write, *24th IEEE-CS Conference on Software Engineering Education and Training,* 371-380, 2011.

7. Parrish A., Dixon, B., Hale, D., and Hale, J., A case study approach to teaching component based software engineering, *13th Conference on Software Engineering Education & Training*, 140, 2000.

8. Runesom, P. and Martin, H., Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering, 14,* 131-164, 2009.

9. Towhidnejad, M. and Aman, J.R., Software engineering emphasis in advanced courses, *ACM SIGCSE Bulletin, 28,* (1), 210-213, 1996.

10. Towhidnejad, M., Salamah, S., and Hilburn, T.B., softwarecasestudy.org, 2011.

BEHAVIOR-DRIVEN DEVELOPMENT*

CONFERENCE TUTORIAL

Dr. Michael Kart
St. Edward's University
Austin, TX 78704

ABSTRACT

Behavior-driven development is a software development technique in which system behaviors are determined and made into test cases before the software itself is written. Benefits can include a richer and deeper understanding of system requirements in addition to making these requirements executable. Moreover, this technique helps guide software developers in knowing what to test as well as knowing how much to test. In this tutorial, we introduce the fundamentals of this approach, a language for expressing system behaviors, and explain how to use this technique effectively in the undergraduate computer science curriculum.

A GENTLE INTRODUCTION TO FUZZY SETS *

Rajan Alex
Department of Engineering and Computer Science
West Texas A&M University
Canyon, TX 79016
806-651-2288
ralex@mail.wtamu.edu

ABSTRACT

The fuzzy set and its applications have been extensively applied in various disciplines. It is suitable in application problems where data is imprecise, historic data is either not available, or cannot be determined exactly. Fuzzy sets maybe used to model problems involving uncertainty. Using fuzzy sets, we can establish relationships that can be used by decision makers to make estimates or predictions in situations involving a great deal of uncertainty, impreciseness and vagueness. Although many such modeling problems using fuzzy sets need a strong foundation in mathematics, the emphasis here is to show how fuzzy sets may be used in simple application problems. Traditionally, very little about fuzzy sets is taught in undergraduate courses. This work establishes the notion of fuzzy sets and fuzzy set operations through examples, and shows how these concepts can be applied in simple real-world application problems for teaching purposes and for understanding the concepts in general.

1. BACKGROUND AND INTRODUCTION

Mathematical modeling is commonly used in engineering, computer science, economics and other disciplines of learning using basic principles derived from mathematics. In mathematical modeling there are some abstractions necessary to transform the problem from a verbal description into mathematical equations. The assumption in such situations is that data is precise, complete and is readily available: however, there are many diverse applications for which it is impossible to get the relevant data. For example, it may not be possible to measure the essential variable in a process such as temperature inside molten glass or the movement of particles in turbulence. There

---

are no measurement scales for some situation variables such as, the smell of a fragrance, the facial beauty of an individual, the roundness of a human face, and so on. Sometimes the data collected for analysis may contain errors. There is a high degree of imprecision in some situations. This may be due to their real-world character and the imprecise relationship that exists between the involved parameters. In such situations, where the mathematical modeling needs exact notions or precise data, mathematically, one can use fuzzy sets for numerical data, temperatures, frequencies, states of the processes, etc., in the traditional mathematical sense of that word. The relationships established using fuzzy sets can be used by decision makers to make estimates or predictions in situations involving a great deal of uncertainty, impreciseness and vagueness.

Fuzzy sets offer a means to represent a gradual change in the actual value of the variable thus allowing error or impreciseness in the data. Fuzzy sets may be used to model a problem when the data is exact. The result of fuzzy set operations in this situation provides another interpretation for the data. This work will introduce the notion of fuzzy sets, show how to decompose fuzzy sets, fuzzy operations, and to defuzzify a fuzzy sets. Further, we propose the use of fuzzy sets as a way to model problems. Through example we incorporate expert's knowledge in a field to model a real world problem. Some of the calculation and derivation in fuzzy sets are fairly complex. However, through examples we will show how fuzzy sets can be introduced in an undergraduate curriculum. There is a very minimal theory of fuzzy sets that is today taught in an undergraduate curriculum. This work shows how one can easily include the concept of fuzzy sets in one's teaching.

The concept of fuzzy sets was first proposed by L.A. Zadeh. A fuzzy set is a class of objects with a continuum [10]. This work about the concepts of fuzzy sets and their operations is a complete introduction and to the best of our knowledge, the descriptions and applications given here are new. However, for further reading on fuzzy sets see [5,7,8,9, and 10]. In section 2, we will introduce fuzzy sets by giving a definition of a fuzzy set. In section 3, we will give descriptions of how to model simple problems and solve it by giving interpretations. In section 4, we will give the conclusion of the work.

2. Membership Degree and Fuzzy Sets

The main concept behind a fuzzy set is that each element in a set is associated with a membership degree, a number in the real interval [0.1]. Let X be a traditional (or non-fuzzy set) of objects called the domain of discourse. According to Zadeh [10], a *fuzzy set* A in X (denoted as) A is defined by a membership function $\mu_A(x)$ which associates with each element $x \in X$ with a real number in the interval [0,1]. Suppose A is a subset of X in the traditional (non-fuzzy) sense then its corresponding fuzzy set is the membership function that takes only two values 0 or 1 with $\mu_A(x) = 1$, if $x \in A$; otherwise 0. In this case, μ_A or A is also called the characteristic function on A. Thus we can say that non-fuzzy sets are also fuzzy sets. If X is the real number line, the graph of a fuzzy set may be represented as in figure 1.

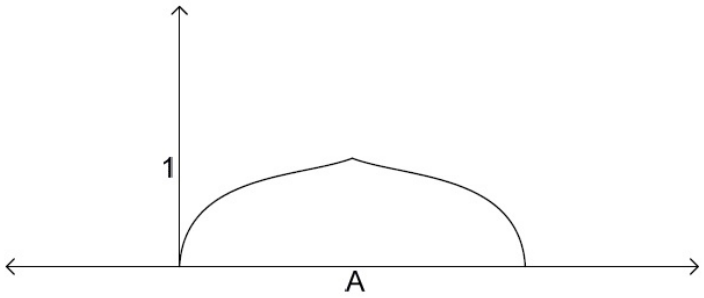


Figure 1 Fuzzy Set A

3. FORMULATION OF FUZZY MODELS

The fuzzy concept can be formulated for any real world problem. The problems for fuzzy modeling are more interesting when the values of the involved parameters are not precise. We will use some basic concepts to model some problems where the exact value of the parameter involve may not be known or cannot be precisely given.

Example 1: Membership function of Circles

Here we try and answer the following question. What does it mean to say that an object is round in shape? It is a linguistic variable. For simplicity, in this example we will look at a round shape in the two-dimensional space. Mathematically, a circle is a set of points that are at a fixed distance from the fixed point. The fixed point is the center and the fixed distance is called the radius of the circle. In the real world, there are objects such as optical discs, the wheels of a car, and so on that are precisely circular objects. These objects are round and represent circles with fixed radii. Now consider objects that may look round in shape such as the face of a person, the circular formation of participants in a school band for a competition, or the shape of an egg. These and many other similar shapes that we encounter in real world are not precisely circular in shape. For example, the round shape of a person's face may be expressed as a fuzzy concept. It may be represented by a fuzzy set CIRCLE with membership function denoted as $\mu_{\text{CIRCLE}}(u)$, where the universe of discussion could be a set of faces (pictures in two dimensions). Each face could be described by a closed curve u. Since a circle minimizes the length l of the curve u with respect to a fixed area a of u, the membership degree may be defined as $\mu_{\text{CIRCLE}}(u) = (\lambda \cdot a)/l^2$, where $a = \pi r^2$ is a real positive parameter. To find λ, we reason as follows: When u is an actual circle $a = \pi r^2$, and $l = 2\pi r$. The quantity a/l^2 and we want the membership degree for a circle u to equal 1. Take $\lambda = 4\pi$. So for an arbitrary non-intersecting closed curve, we define the membership degree as $\mu_{\text{CIRCLE}}(u) = (\lambda \cdot a)/l^2$, and $\lambda = 4\pi$ satisfies $\mu_{\text{CIRCLE}}(u) = 1$ whenever u is a circle. Therefore, the membership function for the round shape of a face u is defined as

$$\mu_{\text{CIRCLE}}(u) = \frac{4\pi a}{l^2}.$$

We can use this function to determine how round is a face curve u by finding the area a and circumference l of the face shape. This function agrees with the definition of a fuzzy set in section 2. Using the concept cut-set of fuzzy set, we can take a threshold value and declare a face round when the membership degree of its shape exceeds the threshold value. Thus, in the example we use the relationship established using a fuzzy set to model a round shape of a face and determine how well round shape is a given face. The following is another example to demonstrate the application of a fuzzy set to a real world situation.

Example 2: The recognition of cancer-cells

Assume that a medical doctor specialist is interested in recognizing cancer-cells. Suppose further that the doctor has a set of variables he is interested in knowing when he examines a patient's blood under a microscope. Let us denote the variable as follows: n_a = area of a nucleus of a cell, n_l = circumference of a nucleus, a = area of a cell, l = circumference of a cell, n_d = optical density of a nucleus, m_d = average optical density of a nucleus, m_t = average transparency in a nucleus. Let us further assume that the doctor has the following rules he has to apply to determine whether the blood sample that he observed under a microscope is cancer infected. The rules are: n_a is too large, n_d is too dark, the ratio of nucleus area to fluids in it is too little, the dye added to the cell does not get evenly distributed in the cell, the nucleus is irregular in shape, and the cell itself is irregular in shape. These rules capture human knowledge. Since there is not one precise number as a value that the doctor can use to infer whether the blood is cancer infected or not, we model each rule as a fuzzy set as follows:

1. n_a is too large. What does it mean to say that a nucleus is too large? It is fuzzy, but we can set "Large nucleus" as a fuzzy concept which can be described by a fuzzy subset A_1 with the fuzzy membership function:

$$\mu_{A_1}(n_a) = \frac{1}{1+\lambda_1(1/n_a)^2},$$

where λ_1 is a real positive parameter. The universe of discussion is the set of n_a for the cells under consideration. We can also say from the statement "nucleus is large" that the linguistic variable nucleus takes the linguistic value large. In general, we can represent fuzzy rules by linguistic variables.

2. n_d is too dark. What does it mean to say that a nucleus is too dark? It is fuzzy, but we can set "Dark nucleus" as a fuzzy concept which can be described by a fuzzy subset A_2 with the fuzzy membership function:

$$\mu_{A_2}(n_d) = \frac{1}{1+\lambda_2(1/n_d)^2},$$

where λ_2 is a real positive parameter. The universe of discussion is the set of n_d for the cells under consideration.

3. The ratio of nucleus area to fluids in it is too little. What does it mean that the ratio of nucleus area to fluids in it is too little? It is fuzzy, but we can set "ratio too little"

as a fuzzy concept which can be described by a fuzzy subset A_3 with the fuzzy membership function:

$$\mu_{A_3}(n_a, n_d) = \frac{1}{1+\lambda_3(n_a / n_d)^2}$$

where λ_3 is a real positive parameter. The universe of discussion is the set of pairs (n_a,n_d) for the cells under consideration.

4. The dye added to the cell does not get evenly distributed in the cell. What does it mean to say that the dye added to the cell does not get evenly distributed in the cell? It is a fuzzy concept, which we can describe by a fuzzy subset A_4 with the fuzzy membership function:

$$\mu_{A_4}(m_d, m_t) = \frac{1}{1+\lambda_4(m_d)^2/(m_d+m_t)^2},$$

where λ_4 is a real positive parameter. The universe of discussion is the set of pairs (m_d,m_t) for the cells under consideration.

5. The nucleus is irregular. What does it mean to say that the nucleus is irregular? It is fuzzy, but we can set "Irregular nucleus" as a fuzzy concept which can be described by a fuzzy subset with the fuzzy membership function:

$$\mu_{A_5}(l, n_a) = \frac{1}{1+\lambda_5(l^2/(n_a-4\pi)^2)},$$

where λ_6 is a real positive parameter. The universe of discussion is the set of pairs (L,n_a) for the cells under consideration.

6. A cell is irregular. Again, what does it mean to say that a cell is irregular? It is fuzzy, but we can set "Irregular cell" as a fuzzy concept which can be described by a fuzzy subset A_6 with the fuzzy membership function:

$$\mu_{A_6}(a, l) = \frac{1}{1+\lambda_6(l^2/a)^2}$$

where λ_5 is a real positive parameter. The discussion universe is the set of pairs (a,l) of the cells under consideration. The above fuzzy membership function definitions use knowledge from the expert and the properties of fuzzy sets. These definitions represent linguistic variables as functions that can be used by decision makers. The cancer recognition is defined as a fuzzy set

$$CANCER = ((A_1 \cap A_2 \cap A_3 \cap (A_4 \cup A_5)) \cup A_6)$$

Using the concepts in section 2, the fuzzy set's membership function is:

$M_{CANCER}U) = max(min\{\mu_{A1}(u), \mu_{A2}(u), \mu_{A3}(u), \mu_{A4}(u), \mu_{A5}(u)\}\}, \mu_{A6}(u)\}$, where $U = \{a, l, n_a, n_l, n_d, m_d, m_t\}$.

In the cancer-cell recognition problem, we can get the membership degree of a cell with respect to the above mentioned indices and calculate $\mu_{CANCER}(u)$. If the membership degree is larger than a threshold, then it needs to be treated as a cancer-cell. A simple and

direct way to compute the parameters in the fuzzy modeling is by consulting an expert in the cancer field.

Example 3: Fuzzy scatter plot

Fuzzy models in application problems that have uncertainty are very common. In such situations, statistical model using probability distribution may be used to model the problem. As an alternative we suggest fuzzy modeling. In fuzzy modeling we have to fuzzify data and also defuzzify resulting model to get result from the available data. A procedure for this is given below.

How to fuzzify?

Assume the data set is a numeric data set. Given a set of data (or imprecise data), $x_1, x_2, ..., x_n$, we can fuzzify them as follows. Rearrange the data in the increasing order of numeric value and re-label them as:

$$A = x_{(0)} \le x_{(1)} \le x_{(2)} \le \ldots x_{(n)} \le x_{(n+1)} = b,$$

Use the triangular fuzzy numbers to define the fuzzy numbers as:

$$\mu_{A_i}(x) = \begin{cases} \dfrac{x - x_{(i-1)}}{x_{(i)} - x_{(i-1)}} & if\ x_{(i-1)} \le x \le x_{(i)} \\ \dfrac{x_{(i+1)} - x}{x_{(i+1)} - x_{(i)}} & if\ x_{(i)} < x \le x_{(i+1)} \\ 0 & otherwise \end{cases} \qquad (1)$$

for $i = 1, 2, ..., n$. Once a set of data is fuzzified as fuzzy sets, fuzzy set operations such as union, intersection, Cartesian product, and so on can be performed on the set to get a fuzzy model for the underlying problem. In this example, from (1), we define the membership function as:

$$\mu_{\tilde{A}}(x) = \frac{\mu_{\tilde{A}_1}(x) + \mu_{\tilde{A}_2}(x) + \cdots + \mu_{\tilde{A}_n}(x)}{n}$$

Then we can obtain a convex membership function as:

$$\overline{\mu}(x) = \cap \{\mu(x) \mid \quad \mu(x)\ is\ convex\ fuzzy\ set, \mu(x) \supseteq \mu_{\tilde{A}}(x)\}$$

The resulting fuzzy set is a fuzzy scatter plot for the given data set. We can also defuzzify to get an interpretation for the data set as follows.

How to defuzzify?

The max-procedure is one of the simplest forms of defuzzificatons. Assume that $\mu_A(x)$ is a fuzzy set and that it takes a single point maximum value y_0. Then defuzzification of μ_A is given as

$$y_0 = \mu_A^{-1}\{\max \mu_A(x) \mid x \in X\} \quad .$$

If there is more than one point in the domain of discourse that attains the maximal value, then we can define:

$$. A_{\max} = \mu_A^{-1}\{\max \mu_A(x) \mid x \in X\} .$$

One way to reach at a defuzzification is to choose one element of A_{MAX} randomly as the defuzzification of μ_A. If one assumes additionally that the domain of discourse X is the set of real numbers, ie., $X \subseteq R$ and that A_{MAX} is a finite set then the defuzzification of the fuzzy set may be defined as follows:

$$y_0 = \frac{1}{N} \sum_{y \in A_{\max}} y ,$$

N is the cardinality of A_{MAX}. An application of fuzzy modeling for a set of data using fuzzification and defuzzification can be found in [2]. Fuzzy modeling problems when data is not know or imprecise that involve the computation of the fuzzy sets may be seen in [1,2,5,6, and 7].

4. CONCLUSION

This presentation provides a basic introduction to fuzzy sets that can easily be taught within an undergraduate curriculum in computer science. Fuzzy sets provide a means to model problems and represent them in a computer even when the data is imprecise or not available. Offered in this paper are some basic operations on fuzzy sets that can be presented to students at the undergraduate level with a minimal level of mathematics background. Fuzzy operations also show that fuzzy sets can interact and form new fuzzy sets. The examples presented here show how to transfer knowledge from the expert into fuzzy sets. A linguistic variable may be used to describe a term or a concept with vague or imprecise value. Through examples, we show how these variables can be represented by fuzzy sets. Fuzzy sets have been extensively used in industry and applications around the world and particularly in Asian countries. Even though it was first introduced in the United States by Prof. Zadeh, its application in the industry is still not very prevalent in the U.S. This work illustrates how fuzzy sets can be used in simple problems, and how inferences may be drawn from them; it also illustrates how these concepts can be taught in introductory courses.

REFERENCES

[1] R.Alex, "Fuzzy Parameters and Their Arithmetic Operations in Supply Chain Systems," in *Supply Chain: Theory and Applications*, Vedran Kordic, Ed. Austria: I-Tech Education and Publishing, 153-176, 2008

[2] S. He, R. Alex, P.Z.Wang "Fuzzy regression based on soft computing," *Intelligent Engineering Systems through Artificial Neural Networks*, Vol.10, 2000, pp. 335-365

[3] D. Dubois, H. Prade, Operations on fuzzy numbers, *International Journal of Systems Science*, 9, 613-626, 1978

[4] D. Dubois, H. Prade, *Fuzzy Sets and Systems: Theory and applications*, Academic press, 1980

[5] Wang, J., Shu, Y.F., Fuzzy decision modeling for supply chain management, *Fuzzy Sets and Systems*, 150 (1), 107-127, 2005

[6] Wang, P.Z., Loe, K.F., *Between Mind and Computer: Fuzzy Science and Engineering*, World Scientific Publishing Co., Inc., 1994

[7] Wang, P.Z., "*Fuzzy Sets and Its Applications*," Shanghai Scientific Press, Shanghai (1983).

[8] Zimmermann, H.J., *Fuzzy Set Theory and Its Applications*, Boston, MA: Kluwer Academic publisher, 2001

[9] R.R.Yager, S.Ovchinnikov, R.M Tong, H.T.Nguyen, *Fuzzy sets and applications - Selected Papers* by L.A Zadeh, John Wiley and Sons, 1987

[10] L.A.Zadeh, "Fuzzy Sets," *Information and Control*, Vol. 8, 29-44, 1969

INTRODUCTION TO ANDROID*

CONFERENCE TUTORIAL

Ken T. N. Hartness
Sam Houston State University

In an effort to engage students with the relevancy of some of the concepts they are learning in our courses, many educators have taken the time to introduce some additional concepts beyond loops and if statements and object-oriented programming. For many young people, the computational device with which they are most familiar is the mobile phone, so introducing mobile programming has been utilized by a number of educators to give programming more exciting relevancy than a checkbook balancing application using text I/O. This workshop seeks to introduce basic concepts of Android programming and engage attendees in a hands-on exercise to develop a simple Android app.

Android programming, as described in the tutorial, requires the Java SDK [1], Eclipse IDE [2] or IntelliJ IDEA [3], and the Android SDK [4]. Additional information about installing and using the software, along with copies of the presentation materials, will be available for a time from http://www.shsu.edu/~csc_kth/android/.

[1] Oracle (2012). Java SE Downloads. Available on 1/16/2012 at http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[2] The Eclipse Foundation (2010). Eclipse IDE for Java Developers | Eclipse Packages. Available on 1/16/2012 at http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/indigosr1.

[3] JetBrains.com (2012). IntelliJ IDEA :: Download Latest Version of IntelliJ IDEA. Available on 1/16/2012 at http://www.jetbrains.com/idea/download/.

[4] Android Open Source Project (2012). Android SDK | Android Developers. Available on 1/16/2012 at http://developer.android.com/sdk/index.html.

Papers of the Fifth Annual CCSC Southwestern Conference

March 23-24, 2012
University of the Pacific
Stockton, California

WELCOME TO THE 2012 CCSC SOUTHWESTERN CONFERENCE

Welcome to the Fifth Annual Consortium for Computing Sciences in Colleges Southwest Regional Conference, held in cooperation with ACM SIGCSE. This year we are hosted by The University of the Pacific in Stockton, California.

We hope that everyone will enjoy Pacific's beautiful campus, expand their peer network and gain some useful ideas as we all continue to strive to improve education in computer science. This year's conference offers invited talks, nine refereed paper presentations, four tutorials, vendor exhibits, and an array of student posters.

The nine papers presented at the conference were selected from twelve submissions, a 75% acceptance rate. The papers were reviewed by a double blind process, with at least three reviewers per paper. The presented papers will be published in the Journal of Computing Sciences in Colleges. On behalf of the conference committee, I would like to extend thanks to all the referees who graciously volunteered their time to help us review the submissions. We would like to give a special thanks to Henry Walker for once again supporting the submission and reviewing process through the software system that he maintains at Grinnell College.

The committee is very happy to welcome and thank our invited speakers. Dan Garcia from the University of California, Berkeley, will present their work developing a pilot for the anticipated new Advanced Placement program in Computer Science. Mehran Sahami from Stanford University will discuss the ACM/IEEE-CS Computing Curricula 2013 effort, as well as curricular changes that have lead to increases in CS enrollment at Stanford. Guy-Alain Amoussou will be discussing opportunities available through the National Science Foundation.

The conference committee and the Southwestern region board did great work preparing for the conference. I'm grateful for their dedication and perseverance, as well as their collegiality and friendship, which made the hard work enjoyable.

The committee is grateful to our hosts at University of the Pacific. In particular, we thank Dean Ravi Jain of the School of Engineering and Computer Science, and Professor William Ford, Chair of Computer Science, whose support made it possible to bring this conference to the Pacific campus. We would also like to thank our national vendors and sponsors: the National Science Foundation, Turing's Craft, Panasonic, and Wiley.

Enjoy the conference and we look forward to your participation at CCSC:SW 2013 at California State University San Marcos.

Michael Doherty
2012 Southwestern Conference Chair
University of the Pacific
Stockton, California

2012 SOUTHWESTERN CONFERENCE COMMITTEE

Michael Doherty, Conference Chair. . . . . University of the Pacific, Stockton, CA
Jinzhu Gao, Site Chair. . . . . University of the Pacific, Stockton, CA
Megan Thomas, Papers Chair. . . California State University Stanislaus, Turlock, CA
Peter Gabrovsky, Authors Chair. . . . . California State University Northridge, Northridge, CA
David Strong, Student Posters Chair. . . . . Pepperdine University, Malibu, CA
Jim Blythe, Speakers Chair. . . . . University of Southern California, Los Angeles, CA
Tzu-Yi Chen, Panels and Tutorials. . . . . Pomona College, Pomona, CA
Youwen Ouyang, Partners and Exhibitors. . . . . California State University San Marcos, San Marcos, CA

CCSC SOUTHWESTERN REGION BOARD OFFICERS

Myungsook Klassen, Region Chair. . . . . California Lutheran University, Thousand Oaks, CA
June Porto, Treasurer and Registrar. . . . . MiraCosta College, Oceanside, CA
Michael Doherty, Editor. . . . . University of the Pacific, Stockton, CA
Colleen Lewis, Regional Representative. . . . . UC Berkeley, Berkeley, CA
Irena Kageorgis, Secretary. . . . . California Lutheran University, Thousand Oaks, CA
Marina S. Doherty, Webmaster. . . . . UC Davis, Davis, CA
Ani Nahapetian, Past Region Chair. . . . . CSU Northridge, Northridge, CA
Stephanie E. August, Past Conference Chair. . . . . Loyola Marymount University, Los Angeles, CA

REVIEWERS — 2012 CCSC SOUTHWESTERN CONFERENCE

Stephanie August. . . . . Loyola Marymount University, Los Angeles, CA
G. Michael Barnes. . . . . California State University Northridge, Northridge, CA
David Bunde. . . . . Knox College, Galesburg, IL
Jimmy Chen. . . . . Salt Lake Community College, Salt Lake City, UT
Tzu-Yi Chen. . . . . Pomona College, Claremont, CA
Mary Jo Davidson. . . . . DePaul University, Chicago, IL

KEYNOTE ADDRESS

Friday, March 23, 2012

THE BEAUTY AND JOY OF COMPUTING (BJC), AP CS PRINCIPLES, AND THE CS 10K EFFORT*

Dan Garcia
Electrical Engineering and Computer Science
UC Berkeley

BJC was chosen as one of the initial pilots for a new "AP CS: Principles" exam to be introduced in 2015. The purpose of this course is to attract nontraditional computing students (especially women and minorities, but also English majors) to the breadth and depth of ideas in modern computer science. The National Science Foundation wants to prepare 10,000 new high school computer science teachers to teach the new AP course by 2015 (the "CS10K" effort). Under their CE21 (Computing Education for the 21st Century) initiative, we were funded to provide paid intensive six-week summer workshops for high school teachers, including two weeks of face-to-face training, one before and one after four weeks of our online course. This talk will review the status of all of these projects, the development of Build Your Own Blocks (BYOB), a graphical programming environment based on MIT's Scratch that is used in the curriculum, and how faculty, students and high school teachers can engage with these important efforts.

DINNER ADDRESS

Friday, March 23, 2012

THE "BIG TENT" OF COMPUTER SCIENCE: CURRICULA FOR THE COMING DECADE*

Mehran Sahami
Stanford University

Interest in Computer Science has fluctuated dramatically in the past 20 years. Many factors have been cited for these enrollment dynamics, including changes in the high-tech economy and the general image of computing. In this talk, we begin by examining some of the factors affecting enrollments in CS, analyzing both historical and current trends. In light of this analysis, we then turn our attention to curricular issues, first examining significant changes made in Stanford University's undergraduate CS program, which aim to expand the scope of education in computer science and highlight the diversity of options available in the field. We discuss the results of these changes -- a near doubling in undergraduate CS enrollments in just two years -- and analyze some of the reasons why. We then look at CS curriculum development more broadly, discussing the ACM/IEEE-CS Computer Science Curriculum 2013 effort (known as CS2013). The goal of this effort is to provide concrete curricular guidance for undergraduate CS programs at the international level for the coming decade.

KEYNOTE ADDRESS

Saturday, March 24, 2012

NSF FUNDING OPPORTUNITIES*

Dr. Guy-Alain Amoussou
Program Officer
National Science Foundation
gamousso@nsf.gov

Guy-Alain Amoussou, is a computer science program director in the Division of Undergraduate Education Directorate for Education and Human Resources. His passion for undergraduate research is fueled by the reward of witnessing how it academically transforms students. His research is interdisciplinary with a focus on design activities related to software systems. He involves his students in two types of undergraduate research. The first type is in senior capstone courses. The second is in the National Science Foundation (NSF) and U.S. Department of Defense (DoD) summer Research Experience for Undergraduates (REU) site. The REU site with the common theme of design is interdisciplinary, involving students and faculty from computer science, mathematics, art/graphic design, and natural resources/Geographic Information Systems. His current scholarship and creative activities include interests in collaborative / interdisciplinary design research, software systems, design knowledge and information assurance; computing education and undergraduate research with a focus on the integration of underrepresented groups.

His teaching included science of design, software engineering, information resource management/knowledge management, programming languages, and telecommunications. Over the past five years, his work was supported by six NSF awards.

Prior to coming to NSF he was a Professor of Computer Science and Director of International Programs at Humboldt State University, a California State University.

DATA MINING FOR STUDENT RETENTION MANAGEMENT*

Shieu-Hong Lin
Department of Mathematics and Computer Science, Biola University
13800 Biola Ave, La Mirada, CA 90639
shieu-hong.lin@biola.edu

ABSTRACT

We conduct a data mining project to generate predictive models for student retention management on campus. Given new records of incoming students, these predictive models can produce short accurate prediction lists identifying students who tend to need the support from the student retention program most. The project is a component in our artificial intelligence class. Students in the class get involved in the entire process of modeling and problem solving using machine learning algorithms. We examine the quality of the predictive models generated by the machine learning algorithms. The results show that some of the machine learning algorithms are able to establish effective predictive models from the existing student retention data.

1. INTRODUCTION

Student retention is a challenging task in higher education [9] and it is reported that about one fourth of students dropped college after their first year [8, 10]. Recent study results show that intervention programs can have significant effects on retention, especially for the first year [7]. To effectively utilize the limited support resources for the intervention programs, it is desirable to identify in advance students who tend to need the support most. In this paper, we describe the experiments and the results from a data mining project in our undergraduate artificial intelligence class to assist the student retention program on campus. The development of machine learning algorithms in recent years has enabled a large number of successful data mining projects in various application domains in science, engineering, and business [4, 11]. In our project, we apply machine learning algorithms to analyze and extract information from existing student data to establish predictive models. The predictive models are then used to identify among new incoming first year students those who are most likely to benefit from the support of the student retention program.

The data mining project serves two purposes. On the one hand, it provides baseline results about the quality of predictive models generated by the machine learning algorithms, which allow the student retention staff to assess the feasibility and utility of incorporating the predictions into the student retention process. On the other hand, it introduces to the artificial intelligence class a real-world application of the machine learning algorithms.

In the remainder of the paper, we describe the details of the project, including the format of the student retention data set, the preprocessing of the data set to protect privacy, the machine learning algorithms applied to the data set, the empirical results on the quality of the predictive models generated by the machine learning algorithms, and the general feedback from the class regarding their experiences in the project.

2. PREPROCESSING THE DATA SET

Students in the artificial intelligence class can access three cleaned-up versions of the student retention data set while participating in the data mining project. They understand the data set is the property of the university and they can not transfer or reveal the data set to others, nor can they use the data set for other purpose. The students also agree to remove the data set from their personal storage after finishing the project.

The raw data set is a collection of 5943 records accumulated over a period of eight years regarding the basic information of first year students and whether they continued to enroll after the first year. Each record in the raw data set keeps track of the values of 52 attributes. In the raw data set, 5009 of the students continued to enroll after their first year while 934 of them dropped out by the end of the first year. We remove attributes involving personally identifiable information such as the name and the student identifier number to protect privacy. We also remove attributes (such as the first-year GPA in college) whose values are not available for new incoming students and are thus useless for student retention prediction. Instead of twelve different attributes regarding various SAT and ACT test scores (many with missing values), we use only a single discretized attribute that summarizes the best test score into one of six discrete levels. Similarly, to make it less likely to be personally identifiable, we only keep discretized attributes that summarize the amounts of financial aid, loan, and scholarship into several discrete levels, and discard the corresponding attributes that record exact amounts. In the end, in the cleaned-up versions of the data set, we only keep twenty two attributes in each record as depicted in Table 1.

Attribute	Description
GENDER	Male or female
ST_RES	State or region from: numeric code
US_CIT	US citizen or not
MAJ_ACAD	Academic major: numeric code
ETH	Ethnic group: numeric code
AGE_c	Age (discretized into 4 levels)
FAFSA	Federal student aid applicant or not
EFCc	Expected family contribution (discretized)
NEED_c	Financial need (discretized)
LOAN_c	Loan received (discretized)
AWD_AMTc	Awarded scholarship (discretized)
CAL_REC	Cal grant receiver or not

Attribute	Description
HOUSE_CoR	Commuter or not
HOUSE_STAT	Living with family or not
HOUSE_ALL	Residence place: numeric code
HS_TYPE	Type of high school attended
GPA_HScnew	High school GPA (discretized)
BES_TESTc	Best test score (discretized)
PER_HSc	Class rank percentile in the high school (discretized)
ACAD_PROB	Under academic probation or not
TYPE_ACPROB	Type of academic probation
RET	Retention: Continue to enroll or not after one year (1 or 0)

Table 1. Attributes used in the cleaned-up versions of the data set

The student retention program needs to focus on students that tend to drop after the first year. However, only less than one sixth of the records in the data set belong to this category. This may mislead some machine learning algorithms to generate models that incline too heavily toward prediction of continued enrollment. To provide a way for balancing such a bias [11], we create two additional versions of the data set by having two or three copies of each of the cases that dropped after a year. Table 2 provides a summary of the raw data set and the cleaned-up versions.

	Description of the contents
Raw data set	5943 records (934 of them dropped after one year, 5009 of them retained), each with 52 attributes.
1x data set	5943 records from the raw data set but only 22 discretized numeric attributes kept.
2x data set	Like the 1x data set, but has 2 copies of each of the original 934 records that dropped after one year.
3x data set	Like the 1x data set, but has 3 copies of each of the original 934 records that dropped after one year.

Table 2. The raw data set and the cleaned-up versions of the data set

3. MACHINE LEARNING FOR FINDING PREDICTIVE MODELS

Weka is open source software that implements a large collection of machine leaning algorithms and is widely used in data mining applications [11]. After learning the conceptual framework of machine learning and the basics of the Weka 3 environment, students participating in the project use Weka to explore the retention data set. They need to conduct experiments on the retention data set to generate predictive models by applying the machine learning algorithms assigned to them. These predictive models (such as those shown in Table 3 below) provide ways to predict whether a new student will continue to enroll or not after one year given the values of the other twenty one attributes.

Simple CART decision tree
derived by the Simple CART algorithm
from the 1x data set

```
GPA_HScnew <= 4
|   NEED_c <= 5
|   |   CAL_REC <= 0      ➔      : 1
|   |   CAL_REC > 0
|   |   |   EFCc_new <= 2 ➔      : 0
|   |   |   EFCc_new > 2  ➔      : 1
|   NEED_c > 5            ➔      : 1
GPA_HScnew > 4            ➔      : 1
```

Alternative decision tree
derived by the alternative decision tree algorithm
(ADT algorithm)
from the 2x data set

```
: 0.493
| (1)GPA_HScnew <= 4        :    -  0.163
| (1)GPA_HScnew > 4         :       0.14
| | (2)BEST_TESTc <= 2      :    -  0.218
| | (2)BEST_TESTc > 2       :       0.135
| (3)NEED_c <= 5            :    -  0.112
| | (4)EFCc_new <= 3        :    -  0.336
| | | (5)NEED_c <= 1        :       0.498
| | | (5)NEED_c >1          :    -  0.384
| | | | (6)EFCc_new <= 2    :    -  0.913
| | | | (6)EFCc_new > 2     :       0.407
| | | | | (7)NEED_c <= 4    :    -  0.927
| | | | | (7)NEED_c > 4     :       0.405
| | (4)EFCc_new >3          :       0.108
| | (10)MAJ_ACAD <= 22      :       0.064
| | (10)MAJ_ACAD > 22       :    -  0.113
| (3)NEED_c > 5             :       0.115
| (8)ST_RES <= 1            :       0.047
| (8)ST_RES > 1             :    -  0.113
| | (9)AGE_c <= 1           :       0.108
| | (9)AGE_c > 1            :    -  0.143
```

J48 graft pruned decision tree
derived by the J48graft algorithm from the 1x data set

```
GPA_HScnew <= 4
|  ST_RES <= 2
|  |  NEED_c <= 5
|  |  |  FAFSA <= -1                          ➔  : 1
|  |  |  FAFSA > -1
|  |  |  |  EFCc_new <= 2
|  |  |  |  |  NEED_c <= 2                    ➔  : 1
|  |  |  |  |  NEED_c > 2                     ➔  : 0
|  |  |  |  EFCc_new > 2
|  |  |  |  |  EFCc_new <= 4
|  |  |  |  |  |  NEED_c <= 3                 ➔  : 0
|  |  |  |  |  |  NEED_c > 3
|  |  |  |  |  |  |  HOUSE_CoR <= 0
|  |  |  |  |  |  |  |  GENDER <= 1           ➔  : 1
|  |  |  |  |  |  |  |  GENDER > 1
|  |  |  |  |  |  |  |  |  HS_TYPE4 <= 3      ➔  : 1
|  |  |  |  |  |  |  |  |  HS_TYPE4 > 3
|  |  |  |  |  |  |  |  |  |  LOAN_c <= 1     ➔  : 1
|  |  |  |  |  |  |  |  |  |  LOAN_c > 1
|  |  |  |  |  |  |  |  |  |  |  HOUSE_STAT <= 2  ➔  : 0
|  |  |  |  |  |  |  |  |  |  |  HOUSE_STAT > 2   ➔  : 1
|  |  |  |  |  |  |  |  HOUSE_CoR > 0          ➔  : 1
|  |  |  |  |  EFCc_new > 4
|  |  |  |  |  |  NEED_c <= 2
|  |  |  |  |  |  |  EFCc_new <= 6
|  |  |  |  |  |  |  |  NEED_c <= 1           ➔  : 0
|  |  |  |  |  |  |  |  NEED_c > 1
|  |  |  |  |  |  |  |  |  EFCc_new <= 5      ➔  : 0
|  |  |  |  |  |  |  |  |  EFCc_new > 5       ➔  : 1
|  |  |  |  |  |  |  EFCc_new > 6             ➔  : 1
|  |  |  |  |  |  NEED_c > 2                   ➔  : 1
|  |  NEED_c > 5                              ➔  : 1
|  ST_RES > 2
|  |  ETH <= 10                               ➔  : 1
|  |  ETH > 10
|  |  |  GPA_HScnew <= 3                      ➔  : 0
|  |  |  GPA_HScnew > 3
|  |  |  |  AGE_c <= 1                        ➔  : 1
|  |  |  |  AGE_c > 1
|  |  |  |  |  MAJ_ACAD <= 24                 ➔  : 0
|  |  |  |  |  MAJ_ACAD > 24                  ➔  : 1
GPA_HScnew > 4                               ➔  : 1
```

Table 3. Examples of predictive models generated by the machine learning algorithms

Table 3 above shows three decision trees as examples of predictive models learned from the retention data set by three machine learning algorithms: the CART decision tree algorithm [4, 11], the J48 graft decision tree algorithm [4, 11], and the alternative decision tree (ADT) algorithm [4, 11]. For example, consider a new case with a high school GPA of 5 (*GPA_HScnew = 5*), best test score of level 2 (*BEST_TESTc = 2*), financial need of level 6 (*NEED_c = 6*), and is an in-state resident (*ST_RES = 1*). For both the CART decision tree and the J48 graft decision tree [4, 11], we need to start from the root to find a unique path leading to a prediction leaf node. In both trees, we find a unique path of length 1 immediately leading us from the root to a leaf node labeled 1, predicting continued enrollment the next year. On the other hand, for the alternative decision tree (ADT tree), we may have multiple paths from the root to the leaves that are consistent with data and we need to sum up all the numbers appearing on these paths to see whether it is positive or negative [4, 11]. In this particular case, we find three paths leading from the root to leaves. Summing up all the numerical numbers appearing on these paths, we have a positive value 0.493+0.14-0.218+0.115+0.047=0.577, and that leads to the prediction of continued enrollment too. These decision trees also provide interesting insights into hidden patterns in the student retention data set. For example, both the ADT tree and the J48 graft decision tree show that age (attribute *AGE_c*) is a very relevant factor only when the student is not an in-state resident (*ST_RES > 1*) or when the student is an international student (*ST_RES > 2*). Not all predictive models can be visualized conveniently like the decision trees in Table 3. For example, a predictive model generated by the naive Bayes algorithm [4, 11] is simply a collection of statistics

derived from the data set while the instance-based nearest neighbor methods [4, 11] essentially match a new case to the entire data set.

In the experiments conducted using Weka 3, we treat all the attributes as numeric attributes and explore the machine learning algorithms applicable to numeric attributes under Weka 3, including (i) fourteen decision tree learning algorithms, (ii) nine decision rule learning algorithms, (iii) four lazy instance-based nearest neighbor algorithms, (iv) seven function-based algorithms for learning neural networks or support vector machines, and (v) five learning algorithms related to the naive Bayes method and Bayesian networks. Typically there are multiple parameters associated with each machine learning algorithm and multiple possible values for each parameter. For each algorithm, we employ mainly the default parameter setting of the learning algorithms in our experiments and have not extensively explored the entire parameter-value space. It is possible that better results can be attained using the same algorithms but with different parameter settings.

4. EVALUATING THE QUALITY OF PREDICTIIVE MODELS

Given the records of new incoming students, we can apply a predictive model to identify students who are likely to drop out if no additional support resources are provided. Staff in the student retention program can more effectively utilize their resources for retention if the predictions are accurate and cover a significant portion of first year students who would drop out if no additional support resources are provided. After a predictive model is established from the student retention data set by a machine learning algorithm, it is then very important to estimate the quality of future predictions generated by the predictive model. In the following, we describe how we conduct cross validation [11] by withholding portions of the student retention data set as test data to evaluate the quality of the predictive model derived by a machine learning algorithm.

Using ten-fold cross validation [11], we first randomly partition the data set into ten subsets, each with 10% of the records in the data set, and then for each subset x we (i) use the algorithm to build a sample predictive model by learning from the other nine subsets combined together and (ii) apply the sample model to predict the retention result for each record in subset *x*. After the cross validation, we can count to find out four numbers regarding the predictions of all the 5943 records in the student retention data set: (i) n_{dd}, the number of correct predictions among those who dropped after first year, (ii) n_{dc}, the number of wrong predictions among those who dropped after first year, (iii) n_{cc}, the number of correct predictions among those who continued to enroll after first year, (iv) n_{cd}, the number of wrong predictions among those who continued to enroll after first year. We then derive two well known measures, precision and recall [4, 11], from these four numbers to estimate the effectiveness of the predictive model (derived from the entire data set by the algorithm) for identifying new first year students who would drop after a year: (i) precision = $n_{dd} / (n_{dd} + n_{cd})$ and (ii) recall = $n_{dd} / (n_{dd} + n_{dc})$. Precision indicates how likely a new case predicted to drop out by the predictive model would actually drop out while recall indicates how likely a would-be drop-out case would be correctly identified by the predictive model.

Table 4 below shows the five machine learning algorithms that produce predictive models with the best precision values in our experiments, together with the

corresponding recall values. There is an obvious trade-off between precision and recall when moving from the 1x version of the data set to the 2x version and the 3x version. For these algorithms, the best precision values (ranging from around 68.8% to 84%) are almost all accomplished when learning from the 1x version of the data set, with recall values ranging from 5.1% to 12.3%. Except for the ADT tree, when learning from the 2x version and the 3x version of the data set instead, the precision of the models drops very significantly to the level from 40% to 56.4% while the recall almost all elevates significantly to around the level 27.8% to 88.7%. However, except for the ADT tree and the NB tree [4, 11], the predictive models learned from the 2x version and the 3x version of the data set are huge decision trees involving one thousand nodes or more, unlike the compact decision trees of at most scores of nodes learned from the 1x version of the data set. Given that we only have around six thousand actual records in the data set, decision trees with thousands of nodes seem to overfit the data set and they may not do well as predictive models for predicting new cases [4, 11].

The alternative decision tree (ADT) learning algorithm is the best precision performer we have seen so far, capable of reaching a precision rate of 84% and a recall rate of 12.4% without a sign of overfitting. In other words, given a collection of 1000 new first year students with around 250 would-be drop-out cases embedded in the list (assuming a drop-out rate of 25% according to [8, 10]), the ADT tree algorithm is likely to produce a list of around 37 students and among them about 31 are actual would-be drop-out cases.

	1x data set		2x data set			3x data set		
	Precision	Recall	Precision	Recall	Over-fitting	Precision	Recall	Over-fitting
ADT Tree	83.9%	12.3%	84.0%	12.4%	Unlikely	49.5%	17.6%	Unlikely
NB Tree	77.9%	07.9%	56.4%	08.9%	Unlikely	40.8%	27.8%	Unlikely
CART	73.8%	05.1%	40.4%	49.0%	Likely	44.9%	88.7%	Likely
J48 graft	70.3%	09.6%	44.4%	35.1%	Likely	43.3%	69.8%	Likely
J48	68.8%	09.9%	43.6%	35.2%	Likely	42.7%	69.8%	Likely

Table 4. Precision and recall accomplished by the top predictive models

5. STUDENT FEEDBACK

The students in the artificial intelligence class responded positively regarding their hands-on learning experiences in the project. Compared with small toy data sets not relevant to them, the students indicated from the very beginning that (i) they saw the value of effective predictive models for student retention management and (ii) they felt curious to see whether machine learning algorithms can learn good predictive models from the student retention data set.

Some students indicated that the project helped them to appreciate the importance of machine learning algorithms and they became interested in learning more about the theoretical foundations of machine learning. For students less interested in the theoretical aspect of machine learning, they liked the broad exposure to the entire data mining process. Many of them would like to explore other data mining application domains in the future and felt that the experiences in this project provided a good foundation for their future exploration.

6. CONCLUSIONS

We see some promising empirical results in the preliminary exploration of the data mining project. Machine learning algorithms such as the alternative decision tree (ADT) learning algorithm can learn effective predictive models from the student retention data accumulated from the previous years. The empirical results show that we can produce short but accurate prediction list for the student retention purpose by applying the predictive models to the records of incoming new students.

The benefits of course projects have been well acknowledged in the general context of education [1, 2] and in specific contexts of teaching AI subjects such as stochastic local search [6], case-based reasoning [3], and hidden Markov models [5]. We believe the data mining project described in this paper is another positive example, demonstrating the value of a student project when teaching the theory and practice of machine learning.

REFERENCES

[1] Barron, B. J., Schwartz, D. L., Vye, N. J., Moore, A., Pertrosino, Zech, A., L., Bransford, J., Doing with Understanding: lessons from research on problem- and project-based learning, *Journal of the Learning Sciences*, 7(3-4), 271-311,1998

[2] Blumenfeld, P. C., Soloway, E., Motivating project-based learning: sustaining the doing, supporting the learning, *Educational Psychologist*, 26(3-4), 369-398, 1991.

[3] Bogaerts, S., Leake, D. , Increasing AI project effectiveness with reusable code framework: a case study using IUCBRF, *Proc. 19th International FLAIRS Conference on Artificial Intelligence*, 2-7, 2005.

[4] Han, J., Kamber, M., Pei, J., *Data Mining: Concepts and Techniques*, 4th Ed. Morgan Kaufmann, 2011.

[5] Lin, S. , An empirical exploration of hidden Markov models: From spelling recognition to speech recognition, *Proc. 19th International FLAIRS Conference on Artificial Intelligence*, 203-208, 2006.

[6] Neller, T. W. , Teaching stochastic local search, *Proc. 19th International FLAIRS Conference on Artificial Intelligence*, 8-14, 2005.

[7] Pan, W., Guo, S., Alikonis, C., Bai, H. , Do intervention programs assist students to succeed in college?: A multilevel longitudinal study, *College Student Journal*, 42, 90-98, 2008.

[8] Tinto, V., *Leaving College: Rethinking the Causes and Cures of Student Attrition*, University of Chicago Press, 1993.

[9] Tinto, V. , *Research and practice of student retention: What next, College Student Retention: Research, Theory, and Practice*, 8(1), 1-20, 2006.

[10] Tinto, V., Russo, P., Kadel, S. , Constructing educational communities in challenging circumstances, *Community College Journal*, 64(1), 26-30,1994.

[11] Witten, I. H., Frank, E., Hall, M. A., *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd Ed. Morgan Kaufmann, 2011.

KNOW YOUR STUDENTS TO INCREASE DIVERSITY: RESULTS OF A STUDY OF COMMUNITY COLLEGE WOMEN AND MEN IN COMPUTER SCIENCE COURSES *

Linda Werner
University of California
Santa Cruz, CA 95064
linda@soe.ucsc.edu

Jill Denner, Lisa O'Connor
ETR Associates
Scotts Valley, CA 95066
jilld@etr.org, lisao@etr.org

ABSTRACT

In this paper, we report on preliminary results of an NSF-funded study of California community college students enrolled in introductory programming courses. There are several unique contributions of our study to computer science (CS) education and social science research. First, it involves large numbers of both women and men from 15 community colleges, allowing us to examine differences in gender, race/ethnicity, and other demographic variables in students' interest and intention to persist in CS. Second, we have collected data on multiple levels of influence: individual, relational, and institutional. Third, we have collected longitudinal data that allows for measuring initial intentions, as well as how experiences in the introductory course change those intentions. We report on several factors that relate to intentions to study CS that can guide interventions to increase diversity.

INTRODUCTION

In the US, the rates of women's enrollment in computer science (CS) courses and completion of CS-related degrees have declined over the last 20 years [32]. Efforts to close the gender gap are limited by a lack of longitudinal and theory-driven research on students' pathways to CS-related degrees, and a lack of research on students who enter the CS pipeline in community colleges (CCs). CCs attract high numbers of women (57% according to the US DoE [41]), and in 2003-4, 32% of all CS-related majors at 2-year public institutions across the US were women, compared to 19% at 4-year public

---

institutions [31]. Recent work reporting on gender differences in associate degrees in STEM fields in the US unfortunately does not include the CS category. Even so, in other non-CS STEM fields, modest gains by women in achieving associate degrees are more than offset by large drops in those achieved by men [22]. One important study [40] reports that both Hispanic or older CS and engineering graduates are more likely than other racial/ethnic groups or younger graduates to have attended a CC. In recent years, partnerships between 2- and 4-year institutions have focused on reducing the structural barriers to transfer, including articulation agreements [20] [30]. Unfortunately, little is known about what motivates CC students to take CS courses or the reasons they do/do not transfer as a CS-related major to a 4-year university. Efforts to bridge CCs with 4-year universities would benefit from this information.

In this paper, we report on the preliminary results of a study that tests three widely held beliefs about why so few women pursue CS-related majors. These beliefs are supported by theory, but there has been little research to support them, particularly among CC students. One widely held belief is that women and men have different levels of motivation to pursue these majors [18] [24]. The second belief is that family support plays a critical role in choice of major, and that parents pressure or socialize their children based on gender stereotypes, a conclusion that is based on research on science and math achievement, but not specifically on CS as an educational choice [8][38]. The third belief is that women's under-representation is due to a lack of previous computer use [9], and specifically a lack of computer game play [10][26]. Our preliminary results are mixed with respect to these beliefs and show them to be a simplification of the reasons for women's under-representation.

THEORETICAL BASIS FOR THE RESEARCH AND RESEARCH QUESTIONS

We base our study on theories that suggest there are individual, relational, and structural factors that can explain gender differences in students' educational pathways. The most common explanations for the under-representation of women in STEM focus on the individual factors, specifically motivation. We draw on three theoretical perspectives on motivation. Eccles' expectancy-value model of achievement-related decisions suggests students are directly influenced by two factors when choosing a college major: the student's expectations for success and the value placed on the course of study [18]. This value is linked to whether they intend to pursue a CS-related career. This model has proven useful for explaining women's educational pathways to non-traditional careers [44], and has been used in a qualitative study to explain their choice of a CS major [39]. In addition, previous studies have found that for women, an interest in programming does not predict intention to pursue the subject if it is not valued [15][21]. Our study is also guided by social learning theory, particularly the concept of self-efficacy, which emphasizes students' beliefs about their ability to perform actions [3]. Self-efficacy beliefs play a central role in the cognitive regulation of motivation, and are found to explain gender differences in interest in math-related college courses [26]. Finally, we draw on self-determination theory, in which motivation is based on students' needs for competence, relatedness, and autonomy [37]. Applications of this theory to education have identified the importance of connections with peers who share an interest in the subject. In particular, women but not men who are CS majors cite peers as one reason they remained in the major [39]. In addition, studies based on self-determination

theory cite the need for supportive autonomy from parents [7]. For example, when students' motivation is self-determined rather than externally controlled, there are higher levels of academic achievement [36].

Our study is also grounded in previous research on the role of the family and peers in the educational pathways of students from under-represented groups. The Bridging Multiple Worlds (BMW) model has been used to describe how expectations of family members, peers, and school personnel influence racial and ethnic minority students in the US [12][14][35]. In contrast to a social capital model in which parent support is positively correlated with academic achievement, Cooper et al [13] described how relationships can be both resources and challenges for students' educational pathways. This view of the family is different from the long-held assumption that more support leads to greater achievement, and is consistent with research on barriers to women pursuing information and communications technology careers [9]. Studies of under-represented minorities in CS-related majors are few, but suggest that Latino/a students are more likely to describe the importance of overcoming family challenges in order to persist in the CS-related major [42], a finding that is consistent with research with Latino/a high school students on the pathway to college [13].

Social learning theory [3] suggests that others' expectations can influence choice of a CS-related major. Based on this theory, many have assumed that gender role stereotypes can explain gender differences in career pathways; however, there is currently no empirical support for this belief. Although traditional gender role expectations by parents and others regarding who is good with computers can undermine girls' access to and confidence with computers [4], these beliefs have not been linked directly to choice and persistence in a CS-related major. One qualitative study found that among CS students, men report higher levels of encouragement from parents than women [39]. A previous study of Canadian students in college science and technology programs found that parent autonomy but not involvement or structure predicted persistence in those fields [24]. Instead of just support, a balance of autonomy and support, particularly for women, to pursue non-traditional educational pathways, might be ideal.

The BMW model suggests that some relational and institutional challenges such as poverty and sexism can motivate students to pursue certain pathways on behalf of their families or to prove others wrong, but only if there are other sources of support such as peer social networks. While studies show how peer groups and feeling comfortable or attached to the college influence student adjustment [1, 2], studies of students in CS classes are limited. Some studies suggest that peer support via the use of pair programming [29] and the presence of same-sex peers have been positive influences on retention in CS-related majors [11], however, it is not known if or how peers influence whether or not under-represented students enter the major. Other institutional challenges, such as a lack of female faculty, may limit women's interest in a CS-related major [6] but may be overcome when students have support from same sex peers [33] or learn to program with a peer [29]. The BMW model has been used to describe pathways to college eligibility and math course taking among under-represented groups, but has not been applied to computer-related outcomes.

There is little understanding of how playing computer games may relate to intent to pursue a CS-related major. Retrospective interview studies have found that male CS majors are more likely than females CS majors to cite an interest in computer games as

a source of motivation and preparation [28][39]. Unfortunately, these previous studies of the role of computer use in choice of major are limited by a lack of theory to guide the research questions and the interpretation of data. One study of middle school girls that drew on Eccles' Expectancy-Value model, as well as self-efficacy theory, found that computer game design is an effective strategy for engaging girls in IT fluency-building activities [17][43]. In particular, students who play certain types of computer games may have more tinkering experience and be more familiar with the cycles of problem solving and failure that are a part of learning to program [17][28]. While playing first-person shooter games has been shown to increase spatial cognition skills, which are important in many math and engineering fields [19], a recent study[33] found frequency of game play was associated with choice of CS major for men, but not for women. This may be because certain types of computer games, such as those that take a long time to learn and master, are more likely to promote the kinds of thinking and problem solving skills that prepare students to succeed in CS classes [16]. Recent data suggest women are using computers and playing games in equal numbers but in different ways [23][25].

Based on the theories described above, findings for groups similar to those studied here, and lack of studies of CC students, this study addresses the following three research questions.

1. What is the relationship between CC student characteristics and CS-related major choice?
2. What are the unique contributions of motivational and familial factors and previous computer use (especially gaming practices) in predicting whether CC students who have shown an interest in computing (i.e., they are enrolled in an introductory computer programming course) declare or intend to declare a CS-related major?
3. Are the contributing factors the same for female and male students and across racial/ethnic groups?

OUR STUDY: METHODS AND PARTICIPANTS

We approached 1723 students at 15 CCs across the state of California at the start of the fall 2010 semester, called time T1, and invited them to participate in a longitudinal study. We chose introductory programming students because they are showing a clear interest in CS. Of the students that were invited, 741 students completed the online survey (43% of the students we reached at T1). We had participation rates of 72% or more at four of the 15 CCs, because some teachers allowed students to complete surveys in lab classes. The online survey measures demographics and students' intentions to pursue a CS-related major at a 4-year university as well as motivational factors (such as value placed on computing, expectations for success with computing and future priorities); familial factors (such as family support, parent occupation and education of maternal and paternal figures); and computer use (such as digital gaming interest and experience and computer use in childhood and now); and other constructs (such as perceptions of sexism, faculty-student interactions, and interactions with other students).

The second data collection period (T2) occurred in spring 2011. We used email reminders and gift card enticements to increase participation. We received T2 surveys

from 583 students, 79% of the students surveyed at T1. The data in Table 1 gives values for students matched at T1 and T2.

Table 1: Descriptive Analyses Results by Gender - the asterisks show where there were significant differences between women and men: *p<0.05, **p<0.01, ***p<0.001.

	women/n=166	men/n=417	range
T1 intent to pursue CS at 4-year univ ***	3.17	3.69	1(definitely not)-5(definitely)
T2 intent to pursue CS at 4-year univ ***	2.85	3.62	1(definitely not)-5(definitely)
Age*	26.16	24.16	15-61 years
Currently employed	55%	47%	
Mom has BA/BS	37%	32%	
White race	49%	49%	
Asian race	37%	32%	
Latino/a ethnicity	19%	20%	
Speaks a language besides English at home at least some of the time	49%	48%	
Grew up only in the USA	70%	72%	
Mom works in computing field*	15%	9%	
Dad works in computing field	20%	16%	
Has an academic degree ***	30%	15%	
Has had a programming mentor*	29%	21%	
Prior programming experience**	31%	43%	
Had a female professor in intro class	31%	31%	
Freq. of game play in a typical week***	2.14	2.87	1(never)-5(every day)

Enrolled in the class to use programming to address social problems	1.68	1.76	1(not at all imp)-3(very imp)
Motivation: value of computers	6.18	6.08	1(strongly disagree) - 7(strongly agree)
Motivation: expectations for success**	3.21	3.35	1(strongly disagree) - 4(strongly agree)
Comfort talking with professor	3.17	3.22	1(strongly disagree) - 5(strongly agree)
Other students encourage me**	1.97	2.26	1(never)-4(often)

Table 1 shows the ways in which the participants are different from the typical 4-year university student who is the focus of most research on pathways to computing careers. Our participants are older, more likely to have a job, less likely to be white (students could check more than one category for race/ethnicity), more likely to have already earned an academic degree, and less likely to have a mother with a bachelor's degree. A summary of the findings, as well as significant gender differences follows.

- Men have a greater intention than women to pursue CS at a 4-year university at both T1 and T2.
- Women are older, more likely to say their mother works in a computing field, more likely to have already earned an academic degree, and to report having had a programming mentor.
- More men than women report having prior programming experience, and they learned it in different ways. Of those that report prior programming, men were more likely to have learned it before college either in a class or by themselves, while women were more likely to have learned it in a college class (48% of the women and 21% of the men learned it in college).
- Men report more frequent computer game play, more encouragement from students in their computing classes to continue in CS-related degrees, and higher expectations for success in computing.

SUMMARY OF RESULTS

We performed hierarchical linear regression analyses predicting student intention to pursue a CS-related degree at a 4-year university at time T2. The following is a summary of the significant results. Unless noted otherwise, these results hold for both genders.

- Students that were not working had a greater intention to pursue computing at a 4-year university than employed students. In our sample, students who were not

working were also significantly younger than those who were (average age of 23.79 years compared to 25.75 years).

- Students who had programming experience before taking an introductory programming class had a greater intention to pursue computing at a 4-year university.
- Students who spent more time playing computer games in a typical week had greater intention to pursue computing at a 4-year university.
- Greater levels of value placed on computing (doing well in CS courses enhance career opportunities, the rewards of a CS degree outweigh the sacrifices, and computers are useful tools) were associated with greater intention to pursue computing at a 4-year university.
- Students who received encouragement from other students in computing classes to continue as a CS major had a greater intention to pursue computing at a 4-year university.
- Men who were more comfortable talking with the professor indicated greater intention to pursue computing at a 4-year university. For women, there was no association between comfort and intention. We did some exploratory analyses to see if the gender of the professor made a difference and found, using simple correlations, that the association between comfort and intention held true for males with male professors, but there was no connection for males with female professors. For women, comfort was not associated with intention, regardless of professor gender.
- Although not quite significant, students who enrolled in an introductory programming course because they wanted to use computer programming to address social problems indicated greater intention to pursue computing at a 4-year university.

The following is a summary of the non-significant results of the regression analyses:

- Having a mentor was not related to intention to pursue a 4-year CS-related degree. This may be due, in part, to the fact that only 23% (48 women/ 89 men) had a mentor at one point in their lives, only 14% (80) still have a mentor at time T2, and only 13 women have had a female mentor. We have additional survey responses about mentoring to explore if there are some conditions that matter.
- Several items addressing the students' experience in the introductory programming course were not significant. The gender of the teacher was not linked to intention for either men or women; perhaps because few of the teachers were female (69% of the participants had a male teacher). The frequency with which the teacher encouraged them to pursue a CS-related major, gave the student advice on how to succeed in CS, or made positive comments about their work was not related to their intention.
- Most family indicators were not significant. Having a mother or father that worked in a computing career did not relate to intention to pursue CS, perhaps because this was true for only 17% of fathers and 10% of mothers. In addition, levels of mother and father support to pursue college, or whether their parents had traditional ideas about gender roles, were not significant.

NEXT STEPS

We have one more student survey planned for Spring 2012. We also plan to conduct additional analyses of data collected at all time points, building on prior research about the barriers and supports for women's pursuit and persistence in CS-related fields. We found that frequency of computer game play was strongly predictive of intention, but we have not explored what it is about game play that makes a difference. We will do this by looking carefully at our data on gaming behavior and preferences to see whether certain types of game play (e.g., mobile, puzzle, action, Internet) are more predictive, and for whom. We also want to explore additional demographic characteristics, since our covariate screening process suggested that students who grew up (at least part time) in a country outside the US had a greater intention to pursue a 4-year CS-related degree. In addition, we plan to look closely at the findings for women from under-represented minority groups, to address these important questions for these populations [34] although our sample sizes will be relatively small. We conducted a survey of the course teachers on teaching methods that are most likely to engage and retain women in CS-related majors [5].

AKNOWLEDGEMENTS

This research is funded by NSF grant 0936791 "Students' Pathways into Computer and Information Sciences Majors: A Study of Community College Men and Women." Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Antonio. A.L. ,The Influence of Friendship Groups on Intellectual Self-Confidence and Educational Aspirations in College, *The Journal of Higher Education* , 75, (4), 446-471, 2004.

[2] Baker, R.W.,Siryk,B.,Measuring adjustment to college, *Journal of Counseling Psychology*,31,1984.

[3] Bandura, A., *Self-efficacy: The Exercise of Control*. NY: W.H. Freeman.

[4] Barker, L., Aspray, W., The state of research on girls and IT, in J.M. Cohoon and W. Aspray (Eds.), *Women and Information Technology: Research on Underrepresentation*, 1-54. Cambridge, MA: The MIT Press. 2006.

[5] Barker, L.J., McDowell, C., Kalahar, K., Exploring factors that influence computer science introductory course students to persist in the major. *ACM SIGCSE Bulletin*, 41,(2), 282-286. 2009.

[6] Beyer, S., Haller, S. Gender differences intragender differences in computer science students: Are female CS majors more similar to male CS majors or female nonmajors?, *Journal of Women and Minorities in Science and Engineering*, 12, 337-365, 2006.

[7] Black, A.E., Deci, E.L. The effects of instructors' autonomy support and students' autonomous motivation on learning organic chemistry: A self-determination theory perspective, *Science Education*, 84, 740-756, 2000.

[8] Bleeker, M.M, Jacobs, J.E., Achievement in math and science: Do mothers' beliefs matter twelve years later? *Journal of Educational Psychology*, 96, 97-109, 2004.

[9] Burger, C.J., Creamer, E.G. Meszaros, P.S. *Reconfiguring the Firewall: Recruiting Women to Information Technology Across Cultures and Continents*, Wellesley, MA: A.K. Peters, Ltd., 2007.

[10] Cassell, J, Jenkins, H., *From Barbie to Mortal Kombat: Gender and Computer Games*, Cambridge, MA: The MIT Press, 1998.

[11] Cohoon, J.M., Aspray, W., A critical review of the research on women's participation in postsecondary computing education, in J.M. Cohoon & W. Aspray (Eds.), *Women and Information Technology: Research on Underrepresentation*, Cambridge, MA: The MIT Press, 2006.

[12] Cooper, C.R., Multiple selves, multiple worlds: Cultural perspectives on individuality and connectedness in adolescent development, in A. Masten (Ed.), *Minnesota Symposium on Child Psychology: Cultural Processes in Development*, Mawah, NJ: Lawrence Erlbaum Assoc., 25-57, 1999.

[13] Cooper, C.R., Cooper, R.G., Azmitia, M., Chavira, G., Gullatt, Y., Bridging multiple worlds: How African American and Latino youth in academic outreach programs navigate math pathways to college, *Applied Developmental Science*, 6, 73-87, 2002.

[14] Cooper, C.R., Denner, J., Theories linking culture and psychology: Universal and community-specific processes, *Annual Review of Psychology*, 49, 559-584, 1998.

[15] Creamer, E.G., Burger, C.J., Meszaros, P.S., Characteristics of high school and college women interested in information technology, *Journal of Women and Minorities in Science and Engineering*, 10, 67-78, 2004.

[16] Denner, J., Bean, S., Girls, games, and intrepid exploration on the computer, in E.M. Trauth (Ed.), *Encyclopedia of Gender and Information Technology*, Hershey, PA: Idea Group Reference, 727-732.

[17] Denner, J., Werner, L. Computer programming in middle school: How pairs respond to challenges, *Journal of Educational Computing Research*, 37, (2), 131-150, 2007.

[18] Eccles, J.S., Understanding women's educational and occupational choices: Applying the Eccles et al. model of achievement-related choices, *Psychology of Women Quarterly*, 18, 585-609, 1994.

[19] Feng, J., Spence, I., Pratt, J., Playing an action video game reduces gender differences in spatial cognition, *Psychological Science*, 18, 850-855, 2007.

[20] Fitzpatrick, J.K-F., Kuehnen, M., Roe, A., Snow, E., Hug, S., *Effective Practices at Community Colleges and Four-year Institutions for Increasing Women in Information Technology Fields*. Colorado Coalition for Gender and IT. http://www.ccgit.org/effectivepractices.html

[21] Goode, J., Estrella, R., Margolis, J., Lost in translation: Gender and high school computer science, in J.M. Cohoon & Aspray (Eds.), *Women and Information Technology: Research on Underrepresentation*, Cambridge, MA: The MIT Press, 2006.

[22] Hardy, D.E., Katsinas, S.G., Changing STEM associate's degree production in public associate's colleges from 1985-2005: Exploring institutional type, gender, field of study, *Journal of Women and Minorities in Science and Engineering*, 16(1),7-30.

[23] Kafai, Y., Heeter, C., Denner, J., Sun, J. (Eds.), *Beyond Barbie and Mortal Kombat: New Perspectives on Gender and Gaming*, Cambridge, MA: The MIT Press, 2008.

[24] Larose, S., Ratelle, C.F., Guay, F., Senecal, C., Harvey, M., Drouin, D., A sociomotivational analysis of gender effects on persistence in science and technology: A 5-year longitudinal study, in H.M.G. Watt & J.S.Eccles (Eds.), *Gender and Occupational Outcomes: Longitudinal Assessments of Individual Social, and Cultural Influences*, Washington, DC: American Psychological Association, 171-192, 2008.

[25] Lenhart, A., Kahne, J., Middaugh, E., Macgill, A., Evans, C., Vitak, J., *Teens, Video Games, and Civics*, at http://www.pewinternet.org/Reports/2008/Teens-Video-Games-and-Civics.aspx, 2008.

[26] Lent,R.W., Lopez,F.G.,Bieschke,K.J., Predicting mathematics-related academic choice and success: Test of an expanded social cognitive model, *Journal of Vocational Behavior*, 42, 223-236, 1993.

[27] Lynn, K.M., Raphael, C., Olefsky, K., Bachen, C.M., Bridging the gender gap in computing, *Journal of Educational Computing Research*, 28, 143-162, 2003.

[28] Margolis, J. Fisher, A., *Unlocking the Clubhouse*, Cambridge, MA: The MIT Press, 2002.

[29] McDowell, C., Werner.L., Bullock,H., Fernald, J., Pair programming improves student retention, confidence, and program quality, *Communications of the ACM*, 49(8), 90-95, 2006.

[30] Morreale, P., Chang, G., Wittenberg, L. Transitioning from a community college to a four-year university, *Computer*, 41, 89-91.

[31] National Center for Education Statistics, *Profile of Undergraduates in U.S. Postsecondary Education Institutions: 2003-2004, With a Special Analysis of Community College Students*. U.S. Department of Education, 2006.

[32] National Science Board. *Science and Engineering Indicators 2006.* http://www.nsf/gov/statistics/seind06/c2/c2s3.htm#c2s312

[33] Ogan, C., Robinson, J.C., Ahuja, M., Herring, S.C. Gender differences among students in computer science and applied information technology. In H.M.G. Watt and J.S. Eccles (eds.), *Gender and Occupational Outcomes: Longitudinal Assessments of Individual Social, and Cultural Influences*, 279-300. Washington, DC: American Psychological Association. 2008.

[34] Ong, M., Wright, C., Espinosa, L., Orfield, G. Inside the Double Bind: A Synthesis of Empirical Research on Women of Color in Science, Technology, Engineering, and Mathematics. White Paper presented to the National Science Foundation, Washington, D.C. (NSF/REESE Project DRL-0635577), March 31, 2010; http://www.terc.edu/work/1513.html

[35] Phalen, P., Davidson, A., Yu, H.C., Students' multiple worlds: Navigating the borders of family, peer, and school cultures, in P. Phelan, A. Davidson (Eds.), *Cultural Diversity: Implications for Education*, New York: Teachers College Press, 1991.

[36] Reeve, J., Self-determination theory applied to educational settings, in E.L. Deci & R.M. Ryan (Eds.), *Handbook of Self-Determination Research*, 183-204, 2002.

[37] Ryan, R.M., Deci, E.L., Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being, *American Psychologist*, 55, 68-78, 2000.

[38] Tiedemann, J., Parents' gender stereotypes and teachers' beliefs as predictors of children's concept of their mathematical ability in elementary school, *Journal of Educational Psychology*, 92, 2000.

[39] Tillberg, H.K., Cohoon, J.M. Attracting women to the CS major. *Frontiers: A Journal of Women Studies*, 26, (1), 126-140, 2005.

[40] Tsapogas, J., The role of community colleges in the education of recent science and engineering graduates, InfoBrief NSF 04-315, Arlington, VA: National Science Foundation, Directorate for Social, Behavioral, and Economic Sciences, 2004.

[41] US DoE. *Descriptive summary of 2003-2004 beginning postsecondary students: Three years later*. National Center for Education Statistics, 2008 http://nces.ed.gov/pubs2008/2008174.pdf.

[42] Varma,R., Prasad,A., Kapur,D., Confronting the "Socialization" barrier: Cross-ethnic differences in undergraduate women's preference for IT education, in H.M.G. Watt & J.S.Eccles (Eds.), *Gender and Occupational Outcomes: Longitudinal Assessments of Individual Social, and Cultural Influences*, Washington, DC: American Psychological Association, 301-322, 2008.

[43] Werner, L., Denner, J., Campe, S., IT fluency from a project-based program for middle school students, *Journal of Computer Science Education Online*, 2, 2005-2006. www.iste.org

[44] Zarrett, N., Malanchuk,O., Davis-Kean,P.,Eccles,J., Examining the gender gap in IT by race: Young adults' decisions to pursue an IT career; J.M. Cohoon and W. Aspray (Eds.), *Women and Information Technology: Research on Underrepresentation*, 1-54. Cambridge, MA: The MIT Press. 2006.

CLASSIFYING PROBLEMS TO EXPLAIN PATTERNS OF CORRELATION ON THE 1988 ADVANCED PLACEMENT COMPUTER SCIENCE EXAM*

Allyson J. Lam, Colleen M. Lewis, Chong (Luke) Lu, Ian B. Ornstein, Dasun Wang
Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

To explore the types of conceptual understanding necessary for novice programmers, we developed a coding scheme for multiple-choice questions from the 1988 AP Computer Science Exam. We used a variety of statistical techniques to compare patterns of students' performance on these questions [3] and overlap in categories for these questions. We found that Bloom's cognitive domain, "Application," was a possible strategy for solving the majority of questions. Additionally, the presence of code in a question was the most significant contributor to correlations between questions. This suggests that perhaps there is a core skill to solving problems that include code, which is independent of the specific content of the code.

INTRODUCTION

Our research attempts to learn more about what computer science concepts and skills are difficult for students. We developed our analysis based upon data presented in Stuart Reges's study regarding the correlations between exam questions on the 1988 Advanced Placement Computer Science (AP CS) Exam [3]. In this study, Reges identified five questions on this exam as "powerhouse questions" that he discovered were correlated most highly with success on the rest of the exam.

Each question in the exam requires students to possess a variety of content knowledge. A high correlation between two questions likely indicates an overlap in the

---

required components for answering both questions. While these components might not be the focus of the question, they play a crucial role in answering the question.

- In analyzing this exam, we investigated the following research questions:
- Which categories were most and least applicable to questions?
- How are the powerhouse questions similar, and how do these questions differ from the questions on the rest of the exam?
- What specific shared components of questions lead to a correlation between the two questions?

These research questions are highly relevant to the teaching and learning of computer science. By exploring the underlying relationship between questions, we can gain a better understanding of how students integrate different concepts within computer science and can create a better understanding of the general cognitive skills required for computer science success. As a result of understanding the skills that are required to succeed in a computing environment, we can better construct our courses to focus on these skills and can increase learning in the computer science classroom. Although technology has evolved tremendously since 1988, the AP CS exam tests introductory computer programming constructs that are relevant to past, present, and future novice programmers.

PREVIOUS RESEARCH

This study builds upon works that investigate hypotheses regarding the nature of programming knowledge. For example, Simon et al. [5] have developed a hierarchical model of programming knowledge and Robins [4] developed a model of the interconnected nature of programming knowledge. Lister et al. [2] argue that the ability to articulate programming knowledge and apply these concepts by tracing through code is fundamental to successfully solving computer science problems. In a statistical analysis of the 1988 AP CS Exam, Reges [3] discovered a few multiple-choice questions that had significant predictive value for students' success on the entire exam. These "powerhouse questions" he asserted, tested some critical component of computer science understanding. Reges hypothesized that there are underlying skills that enable students to understand and solve a diverse set of computer science problems. In our research, we developed a coding scheme to identify the different skills and knowledge used in the 1988 AP CS Exam questions studied by Reges [3]. Sheard et al. [6] created a classification scheme that could be used to analyze the focal content and trends of computer science examinations and in line with their project goals chose to limit their number of categories. To categorize a more comprehensive set of skills tested in 1988 AP CS Exam, we used the original Bloom's taxonomy [1] as a starting point. Bloom's taxonomy has been applied to Computer Science before [7]. These authors [7] interpreted previously defined categories in the revised taxonomy through the lens of computer science, identified verbs and keywords that clearly indicated specific categories, and provided exam questions that fit into each of Bloom's cognitive domains. This study provided an initial basis for our analysis of the AP CS questions, as many of the questions we explored were similar in content and wording to the examples discussed in this study.

METHODS

Methods: Development of Categories

We developed a set of coding categories through an iterative process. Our initial set of categories included a list of topics covered on the exam and Bloom's taxonomy. With this initial set of categories, four coders independently coded sets of ten to fifteen questions. We resolved discrepancies through discussion and noted changes to the category definitions in an evolving document. These discussions frequently resulted in the suggestion to split an existing category or create a new category. We discussed these proposals and made changes to the category definitions. We repeated this process of coding questions and revising categories until all questions from the 1988 exam had been coded and all discrepancies had been discussed and resolved. This process resulted in the development and refinement of 44 categories.

Methods: Final Coding of All Questions

To obtain higher coding consistency, two researchers re-coded each question. Any discrepancies among these coders and the previous coding of the question were discussed and resolved by the entire research team. Our raw data from the coding consists of a matrix recording the final coding for each question and category. All categories were coded with a binary value, with a 1 if the question was coded with that coding category and a 0 otherwise.

Methods: Occurrences of Codes Across Questions

For each category, we created a table that contained (1) the percentage of powerhouse questions that we coded as that category and (2) the percentage of non-powerhouse questions we coded as that category. This analysis allowed us to investigate the hypothesis that there were certain categories that played a role in a majority of the questions, while other categories applied to only a small portion. It also let us compare the frequency that a powerhouse question, versus a non-powerhouse question, was coded with a particular category. This in turn enabled us to investigate which questions were similar to and which questions differed from the five powerhouse questions.

Methods: Similarity of Coding

In an effort to quantitatively determine how similar questions were, we compared the coding for every pair of questions. To determine a percentage of how similar each pair of questions was, we divided the number of similar codings between the two questions by the total categories coded by either question. The similar codings represented how many times both questions were coded with '1' for the same category. We calculated the total categories coded as the total number of categories coded by either question.

Methods: Least-Squares Analysis

Reges [3] provided data for the correlations between powerhouse questions and the other questions that had correlations above 0.2. These data formed the basis for the following analysis: For each of the 5 powerhouse questions, we formed a new array, which we will refer to as the Category Overlap Array. Each Category Overlap Array expressed how related each question was to the powerhouse question that formed the array.

Each Category Overlap Array had one column for each of the 44 coding categories. Each row represented a question that was non-trivially correlated to the powerhouse question that formed the array. An entry in the Category Overlap Array was a 1 if the particular powerhouse question that formed the array and the question indicated by that row both shared that particular coding category and 0 otherwise. Therefore, each of the five Category Overlap Arrays contained information concerning the similarities between each question and the powerhouse question that formed the array.

We conducted five least-square calculations to find five weight vectors that represented how strongly each coding category determined the correlation between the powerhouse question that formed the array and another question. This mathematical process finds a vector of values x, the weight vector, that best fits the equation A*x = b, where A is a matrix, in this case the Category Overlap Array, and b is a vector of the correlations between the powerhouse question that formed the array and other questions, as given by Reges [3].

All of our calculated weight vectors had entries that were N/A; this happened when two questions did not share a coding category and therefore it was not possible to figure out the correlation determined by the coding category.

After determining the weight vector for each powerhouse question, we averaged these values by summing all the weights and dividing by the number of non-N/A entries for that coding category. We averaged across multiple questions in order to estimate the coding category's overall importance in determining correlation for all questions with the data provided.

RESULTS

Results: Occurrences of Codes Across Questions

We discovered that each powerhouse question was coded as Bloom's: Application. Over 50 percent of all questions were coded as Bloom's: Application. This shows that the ability to trace through code with specific values is a valuable skill, which a student could use to solve more than half of the questions, including every powerhouse question.

However, some questions had multiple problem solving approaches. For example, the question most highly correlated with success, Question 23 ("If *b* is a Boolean variable, then the statement *b:= (b = false)* has what effect?"), can be solved by tracing through the question with specific values for b, indicative of *Bloom's: Application.* However, a student could also inspect the question and deduce that the first half of the question *(b :=)* is doing the variable assignment, while the second part *(b = false)* is doing the comparison. Using that, a student could see that the second phrase *(b = false)* will always result in the opposite value of what *b* is, so setting b to that value *(b :=)* will

always change the value of *b*. In that regard, a student would be able to solve the problem without tracing through specific values, in this case indicative of *Bloom's: Understand*. This double-coding allowed us to incorporate multiple ways to approach the problem.

Results: Similarity of Coding

None of the powerhouse questions were within the five questions with the highest average level of similarity to other questions on the exam. This demonstrates that the most correlated questions identified by Reges [3], were not those questions that shared the greatest number of features with questions across the exam. This is consistent with our hypothesis that some features may play a larger role in the correlation between questions. The average level of similarity for all pairs of questions was 16.1 percent. The question most highly correlated with success on the 1988 AP CS Exam, Question 23, was on average only 16.3 percent similar to other questions. Questions coded as *Bloom's: Application*, the most frequently used category, had a higher average level of similarity (30.5 percent), than the average similarity for all questions that were not coded with Bloom's: Application (11.4 percent). We expect that this similarity came from a pattern of coding a number of categories frequently associated with *Bloom's: Application,* such as *Array, Integers, Loops, Implicit/Explicit need to trace code*, and *Has code in question*.

Results: Least-Squares Analysis

In many cases, only one powerhouse question provided information about the weight of a coding category. In addition, the omission of weakly correlated questions limited our analysis in determining which categories did not lead to correlation. Therefore it is best to qualitatively examine these numbers.

We divided these weights at natural divisions in the data at 0.1 and 0.01 to group coding categories into clusters that we describe as having a "strong correlation," "mild correlation," or "weak correlation." Also note that some of these categories have negative correlations. This does not suggest that this coding category creates an inverse relationship between two questions. Because of the nature of our analysis, negative numbers act as the smallest calculated correlations. Table 1, below, shows the average weight vectors for the 19 coding categories we were able to determine with the data provided.

Table 1: Correlations of Coding Categories

Strong Correlations (>0.1)	**Mild Correlations (0.1>0.01)**	**Weak Correlations (<0.01)**
Has Code in Question (0.207) *Array* (0.190) *Linked List* (0.153) *Compiler* (0.144) *Recursion* (0.112)	*If/else* (0.0941) *Bloom's: Evaluation* (0.0700) *Implicit need to trace code* (0.0431) *Bloom's: Application* (0.0424) *Bloom's: Comprehension* (0.0389) *Execution time* (0.0300) *Bloom's: Recall* (0.0215) *Run-time errors* (0.0138)	*Explicit need to trace code* (0.00533) *Describe code* (-0.00479) *Multiple part question* (-0.0147) *Booleans* (-0.0451) *Invariants* (-0.106) *Type definition is given* (-0.166)

Our highest correlation by far came from questions that shared the coding category: *Has Code in Question*. Not only was this correlation number far higher than any of the others, but it had a high correlation factor for four of the five powerhouse questions. This suggests that perhaps there is a core skill to solving problems that include code, which is independent of the specific content of the code.

Four of the highest correlated coding categories are concerned with specific topics in the question *(Arrays, Linked Lists, Compiler,* and *Recursion)*. The implication is that if a question asks about *Arrays, Linked Lists, Compiler*, or *Recursion*, then knowledge of that topic lies at the heart of the question. This might be expected: it is intuitive to think the data structures or computer science topics a question focuses on determines a student's ability to answer the question. Accordingly, categories involving topics or data structures accounted for only 25 of the 44 coding categories, but amounted to four out of five strongly correlated categories.

We should not make the fallacy that a coding concept creates a correlation between two questions, if both of those questions ask about the concept. Instead, a coding concept that indicates a high correlation between two questions suggests that the concept is an essential part of both questions.

LIMITATIONS AND THREATS TO VALIDITY

Validity of Categories and Coding

Although we created thorough definitions for each of our coding categories, we acknowledge that these categories do not capture elements of each exam question in a uniform way. For example, we identified three reasons a question might be coded as Recursion: "(1) Code includes a recursive procedure that must be understood in order to answer the question correctly. (2) Calls its own method. (3) Question asks about a recursive procedure." Most of the questions we coded as Recursion qualified for this category because they included recursive procedures. However, due to the last statement in the coding definition, we also coded questions as Recursion if they even mentioned recursion in the answer options. Like Recursion, most of our categories did not distinguish between questions that involved a given category as an essential component and questions that simply mentioned that category. This means that two questions using a concept in two completely different ways could be coded under the same category; however, this also gave us a more manageable number of categories, which were easier to classify questions with. Previous research [6] has chosen to limit the number of categories that can be applied to a particular question. However, we expected that the less prominent content in a question might be responsible for the correlations across questions. Therefore, we chose to code primary and secondary content in the questions.

When it was possible to solve a problem in multiple ways, we coded the question with codes that were relevant to any of the possible strategies. However, we may not have thought of every possible strategy students could use to solve each problem, which impacts our coding scheme's accuracy.

Inter-rater Reliability

During our second coding of all questions, we divided the questions between two pairs of coders; each pair re-coded half of the exam. We discussed any discrepancies between the previous consensus and individual coders. If either person's coding disagreed with the final consensus, we counted that as one error. If both coders originally disagreed with the final consensus, we counted that as two errors. We calculated our errors across our entire coding matrix and found that individual coders were consistent with the final coding 95.43 percent of the time.

CONCLUSION

Previous work to classifying computer science questions with Bloom's Taxonomy [7] provided a basic framework for our study, as we examined computer science problems and distinguished key problem solving abilities. Through our coding we identified patterns across questions on the 1988 AP CS Exam. It was particularly interesting that a majority of questions on the exam, including all five of the powerhouse questions, were categorized as Bloom's: Application. The prevalence of this category suggests the usefulness of being able to trace through code with specific values, and supports previous research that emphasizes the ability to trace code as a demonstration of thorough programming understanding [2].

Our analysis also shows that the category *Has code in question* has the strongest correlation between powerhouse questions and questions with non-trivial correlations to powerhouse questions. This may suggest a qualitative difference between questions that have code and questions that do not and it is consistent with the frequency of identifying questions as *Bloom's: Application*. The powerhouse questions, which according to Reges [3] best predicted success on the exam, all had code in them. The implication is that tracing through code is a difficult and vital skill for success in the 1988 AP CS Exam. There were also high correlations for specific topics in computer science, including *Array, Linked list, Compiler,* and *Recursion*.

In our future research, we will be analyzing student data from the 2004 and 2009 AP CS exams to investigate if the pattern of highly correlated powerhouse questions is replicated on later exams. We will extend the analyses of coding presented here by using student data rather than pre-computed correlations as we used from Reges [3].

REFERENCES

[1] Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H. and Krathwohl, D.R. (1956) *Taxonomy of educational objectives Handbook 1: cognitive domain*. London, Longman Group Ltd.

[2] Lister, R., Adams, E., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Mostrom, J., Sanders, K., Seppala, O., Simon, B., Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE*, 36, 7-10.

[3] Reges, S. (2008). The mystery of b := (b = false). *ACM SIGCSE*, 39, 21-25.

[4] Robins, A. (2010). Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education*, 20(1), 37-71.

[5] Simon, Cutts, Q., Fincher, S., Haden, P., Robins, A., Sutton, K., Baker, B., Box, I., de Raadt, M., Hamer, J., Hamilton, M., Lister, R., Petre, M., Tolhurst, D., Tutty, J. (2006). The ability to articulate strategy as a predictor of programming skill. Proc Eighth Australasian Computing Education Conference, Hobart, Australia, Jan 2006.

[6] Sheard, J., Simon, Carbone, A., Chinn, D., Laakso, M., Clear, T., deRaadt, M., D'Souza, D., Harland, J., Lister, R., Philpott, A., Warburton, G. (2011). Exploring programming assessment instruments: classification scheme for examination questions. *ICER 2011*.

[7] Thompson, E., Luxton-Reilly, A., Whalley, J., Hu, M., Robbins, P. (2008). Bloom's taxonomy for CS assessment. *ACE2008*.

SNAP! (BUILD YOUR OWN BLOCKS)*

TUTORIAL PRESENTATION

Dan Garcia
Electrical Engineering and Computer Science
UC Berkeley
ddgarcia@cs.berkeley.edu

Luke Segars
Electrical Engineering and Computer Science
UC Berkeley
lukes@cs.berkeley.edu

Josh Paley, Computer Science Teacher
Henry M. Gunn High School in Palo Alto
josh.paley@gmail.com

ABSTRACT

This workshop is for high school and college teachers of general-interest ("CS 0") computer science courses. It presents the programming environment used in two of the five initial AP CS Principles pilot courses.

SNAP! (Build Your Own Blocks) is a free, graphical, drag-and-drop extension to the Scratch programming language. Scratch, designed for 8-14 year olds, models programs as "scripts" without names, arguments, or return values. SNAP! supports older learners (14-20) by adding named procedures (thus recursion), procedures as data (thus higher order functions) structured lists, and sprites as first class objects with inheritance.

Participants will learn SNAP! through discussion, programming exercises, and exploration. See http://snap.berkeley.edu for details. Laptop required.

INTENDED AUDIENCE

High school and undergraduate computer science teachers.

PRESENTERS' BACKGROUND/BIOGRAPHY

Dan Garcia is a Lecturer with Security of Employment in the EECS Department at the University of California, Berkeley. He received his Ph.D. in Computer Science from UC Berkeley in 2000. He was the co-developer and co-instructor (with Brian Harvey) of Berkeley's Advanced Placement Computer Science: Principles pilot course CS10: The

---

Beauty and Joy of Computing. He is on the Advanced Placement Computer Science: Principles Advisory Board, and the ACM Education Board.

Luke Segars is a masters student at UC Berkeley studying computer science and its potential impacts on education, particularly as a tool for motivating student interest. Luke is also the head teaching assistant for the SNAP!-based "Beauty and Joy of Computing" course at Berkeley. He has also used Scratch, CS Unplugged and other tools to teach students at the elementary, middle, and high school levels.

Josh Paley has been teaching computer science at Gunn High School, Palo Alto, for ten years. He studied computer science at the University of California, Berkeley. He was among the first high school teachers to pilot the SNAP!-based "Beauty and Joy of Computing" curriculum, used at two of the initial five pilot sites for the coming AP CS Principles course.

ROUGH AGENDA FOR THE TUTORIAL

Note: This agenda takes into account the feedback from participants in last year's SIGCSE workshop that it was overscheduled with lecture, and not enough time for participants to develop their own projects. We have therefore scheduled 40 minutes of presentation and 50 minutes for a combination of suggested exercises and free project development time.

20 minutes: Introduction to AP CS Principles, SNAP! Building your own blocks and recursion
10 minutes: Programming time
10 minutes: First class data types, lambda, and higher order functions
20 minutes: Programming time
10 minutes: Prototype-based object oriented programming
20 minutes: Programming time

AUDIO-VISUAL AND COMPUTER REQUIREMENTS

We need just a projector and screen. Participants will bring their own laptops.

OTHER CRITICAL INFORMATION

SNAP! (formerly known as BYOB) is the graphical programming language used in two of the five phase 1 pilot sites for the coming AP CS Principles exam. We are undertaking an NSFfunded outreach and teacher preparation project this year and expect widespread interest at CCSC:SW.

THE IMPACTS OF PROVIDING NOVICE COMPUTER SCIENCE STUDENTS WITH A SECOND CHANCE ON THEIR MIDTERM EXAMS*

Ben Stephenson
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta T2N 1N4
Canada
ben.stephenson@ucalgary.ca

ABSTRACT

Over the years, we have been approached by numerous students seeking a "second chance" after performing poorly on a midterm exam in an introductory computer science course. While many students have provided creative justifications for why they should be treated differently from everyone else, some students have articulated reasons why all students in the class should receive a second chance. These reasons have included lack of experience writing university level exams, the cumulative nature of the final exam, and lack of prior experience writing exams in this subject area.

After considering the arguments, the instructor for the course adjusted the grading scheme in a subsequent year so that students would receive a second chance on the midterm exam. Specifically, students who achieved a higher grade on the final exam than on the midterm exam would have their midterm exam grade replaced with their final exam grade. Making this change resulted in improved student perception of the fairness of the grading scheme while having little impact on the class average grade for the course. While these positive outcomes were observed, some negative impacts also occurred including lower midterm exam participation rates and a higher failure rate for the course.

1.0 INTRODUCTION

Course outlines often describe learning outcomes that students are expected to achieve by the end of the course. Yet instructors often determine a significant portion of students' grades at the midpoint of the academic term using a test or exam. Locking in a significant portion of students' grades at this point of the term fails to reward students who master this material later in the course. During our teaching career, we have also had several students express that they believe their mark on the midterm exam didn't accurately reflect their knowledge because of lack of experience writing either university level and/or computer science exams.

Seeing some validity in these arguments, we decided to provide students with a second chance on the midterm exam. This second chance was implemented by replacing students' midterm exam grades with their final exam grades when the final exams showed improved performance. Our hope was also that this would change the midterm exam from being purely an assessment tool into a combination of an assessment tool and a learning experience.

The remainder of this paper is organized in the following manner. Section 2 provides a brief overview of the students taking the course, the topics covered in the course, and the students' expected level of understanding. Section 3 describes the process we followed to assess the impact of our change in grading practice. The differences we observed in students' level of participation in various assessments and students' grades are described in Section 4. It also outlines changes observed in the course evaluations completed by the students. Section 5 presents our conclusions.

2.0 COURSE DESCRIPTION

At our institution the Faculty of Science offers a Natural Sciences Program. This program is targeted at students "who have a broad interest in science and wish to emphasize the fundamental importance of a multidisciplinary approach" [5]. As part of this program, students are required to take courses from at least four departments within the faculty of science, including at least one half-course from the department of computer science with significant programming content. As students advance in their program, they select two concentration areas where they take more advanced courses.

Our introductory computer science course serves multidisciplinary students, many of whom are enrolled in the Natural Sciences Program. As such we have students who are scientifically and mathematically literate, but that are not necessarily inherently interested in computer science. Separate courses are offered for students majoring in computer science programs, and a separate course with little programming content is offered to support programs outside of the faculty of science.

The overall goal of our course is to provide students with the skills necessary to write small computer programs that will assist them in their pursuit of knowledge in other scientific disciplines. Upon completing the course, students are expected to be able to take a description of a problem, decompose it using top down design, and then synthesize a solution to the problem as a Python program. During the course students gain experience using common programming constructs such as if statements, loops and functions. They are introduced to array and dictionary data structures which they must

use to complete the final two assignments in the course. By the end of the course they are also expected to be able to load data from files and perform basic exception handling. Throughout the course students make use of a graphical package [4] which allows them to perform simple visualizations such as graphing a dataset.

3.0 EXPERIMENTAL DESIGN

We have taught introductory computer science to multidisciplinary students on several occasions. In fall 2008 we taught the course for the first time using materials inherited from the previous instructor. We made some adjustments to the course content and delivery before teaching the course for a second time in spring of 2009. A few additional adjustments were made between the spring 2009 and fall 2009 terms. Since fall 2009 the course has remained stable, both in terms of its content and the techniques used to deliver the content.

In this study we compare data from the fall 2009 academic term with data from the fall 2010 academic term. The course was a large enrollment class each year, with 150 students in 2010 and 180 students in 2011. Two sections of the course were offered each year, all with the same instructor. All of the comparisons performed in this study consider the entire year's data. We don't attempt to perform any analysis for individual sections because we noticed that some students choose to attend the earlier lecture section even though they are actually enrolled in the later lecture section, and vice versa. The class times were the same each year.

The instructor made a conscious effort to minimize differences in the course between these terms. Students were recommended to purchase the same textbook each year, and students were provided with nearly identical course notes (a handful of typos were corrected from 2009 to 2010). The instructor did not consciously change any aspects of the course beyond the grading policy that is the focus of this study.

Students completed the same number of assignments each year. While the assignments were not identical, they were highly similar. Assignment 1 included a small programming task along with a collection of written questions about data representation. Students were asked to complete the same programming task each year. The data representation questions used slightly different numbers in 2009 and 2010.

Later assignments were adjusted in similarly small ways. In 2009, assignment 2 asked students to create a program that plotted an equation of the form $y = ax^3 + bx^2 + cx + d$. In 2010 students created a program that plotted an equation of the form $y = a(x-b)^c + d$. Assignment 3 asked students to implement four image transformations. In 2009, the operations were darken, chroma key, rotate by 180 degrees and blur. In 2010 the operations were negative image, chroma key, rotate 90 degrees counterclockwise and blur. Assignment 4 asked students to read data from a file, process it, and generate a bar graph. No significant changes were made to this assignment between 2009 and 2010.

Students were also assessed using a midterm exam and a final exam. Because the midterm exam was returned to the students it was necessary to make significant changes to it between 2009 and 2010. As such, we do not perform any comparisons on midterm exam results between the two years.

Final exams were not returned to students in either year. While students had the opportunity to review their final exam after final marks for the course were posted, no students elected to utilize this opportunity. As such, we were able to use the same final exam in both 2009 and 2010. The exams were carefully controlled between 2009 and 2010 to ensure that no students gained access to them.

4.0 RESULTS

We examined the impact our change in grading policy had on a variety of factors. Section 4.1 examines changes in the number of students that elected to participate in each assessment. The number of students that withdrew from or failed the course is explored in Section 4.2. The impact this change had on final exam grades is described in Section 4.3 and Section 4.4. Section 4.5 describes the impact this change had on the class average grade. Course evaluation data is presented in Section 4.6.

4.1 Participation Rates

We examined the proportion of students that submitted each assessment each year. Table 1 summarizes the behavior we observed.

	2009	2010	Difference	Chi Square
Number of Students (n)	150	180		
Assignment 1	90.0%	91.7%	1.7%	0.56
Assignment 2	86.0%	83.9%	-2.1%	0.67
Assignment 3	79.3%	78.9%	-0.4%	0.02
Assignment 4	71.3%	73.9%	2.6%	0.57
Midterm Exam	96.0%	90.6%	5.4%	13.89
Final Exam	80.7%	81.7%	1.0%	0.12

Table 1: Assessment Participation Rates in 2009 and 2010

Table 1 shows that most of the participation rates were similar each year. A Chi Square (?2) test of proportions was applied to each assessment to determine if the differences observed were statistically significant. The differences for the assignments and the final exam were not found to be statistically significant at a 95% confidence level (critical value: 3.84).

A statistically significant difference was observed in the Midterm Exam participation rate (critical value: 3.84, observed value: 13.89). In 2009, 144 of 150 students (96.0%) wrote the midterm exam for the course. In 2010, the midterm exam participation rate dropped to only 90.6% (163 of 180 students).

This difference in midterm exam participation rate is worrisome. When students fail to write the midterm exam 70% of the students' final grades are determined from a single 2-hour assessment. This is worrisome, both because students gave up the

opportunity to "lock-in" a portion of their grade by writing the midterm exam, and because, on average, students do more poorly on the final exam. We believe that this difference resulted from some students taking the midterm exam less seriously because they knew that they would have a second chance to earn those marks. Walker has previously noted that "Student's typically view points as identifying what matters" [7]. The same conclusion is also reached by Gibbs and Simpson [3]. In this case, our grading policy may have inadvertently sent the message that the midterm didn't matter since choosing not to write it did not result in the direct deduction of any points.

4.2 Course Failures vs. Course Withdrawals

At our institution, students are able to withdraw from a course until the last day of classes. When a student withdraws from a course a grade of W is recorded in the student's academic record. However this W does not impact the student's grade point average. When a student fails a course a grade of F is recorded in the student's academic record. A grade of F is included in a student's grade point average, contributing 0 grade points. As such, students benefit from withdrawing from courses instead of failing them.

As part of this study, we examined the pattern of student failures and withdrawals. Table 2 shows the distribution of withdrawals and failures each year.

	2009	**2010**	**Difference**
Number of Students (n)	150	180	
Passed	72.7%	72.8%	0.1%
Withdrew	18.0%	13.3%	-4.7%
Failed	9.3%	13.9%	4.6%

Table 2: Student Failures and Withdrawals

While the proportion of students who didn't receive credit for the course was nearly identical in each year (27.3% in 2009, 27.2% in 2010), more students failed the course in 2010, with the resulting negative impact on their grade point averages. Using a Chi Square test of proportions revealed this difference was statistically significant (critical value: 5.99, observed value: 6.18).

We believe that this difference resulted from students having unrealistic optimism about their ability to achieve a substantially better grade on the final exam than on the midterm exam, and that this optimism lead them to write (and fail) the final exam when withdrawing from the course would have been the more prudent choice. This phenomenon is known as optimism bias [1], and can occur both inside and outside the academic arena [2].

4.3 Final Exam Grade Improvement

Of the 147 students that wrote the final exam in 2010, 28 earned a higher grade on the final exam than on the midterm exam. As such, approximately 19.0% of students improved their grade as a result of this change. For comparison purposes, 102 students

(69.4%) achieved a lower grade on the final exam than on the midterm exam (and as such, retained their midterm exam grade), and approximately 11.6% of students achieved the same grade.

The amount of improvement varied greatly among students that achieved a higher grade on the final exam than on the midterm exam. While many students only improved by a third of a letter grade (for example, moving from a C to a C+), one student improved from a D+ on the midterm exam to an A on the final exam. On average, students that improved increased their grade by approximately 0.73 grade points. The following figure shows the distribution of the improvement.

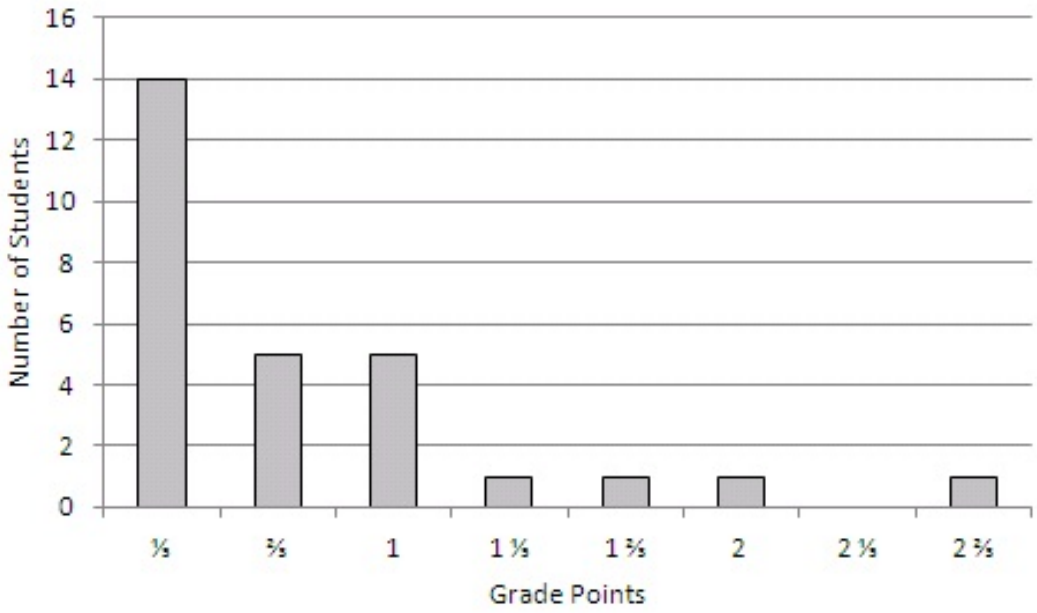


Figure 1: Amount of Improvement among Students that Achieved a Higher Grade on the Final Exam

4.4 Which Students Showed Improvement?

As noted previously, approximately 19.0% of students improved their midterm exam grade by earning a higher grade on the final exam. In this section, we examine which students benefited from this opportunity. Figure 2 shows the proportion of students that improved on the final exam, divided by midterm exam grade.

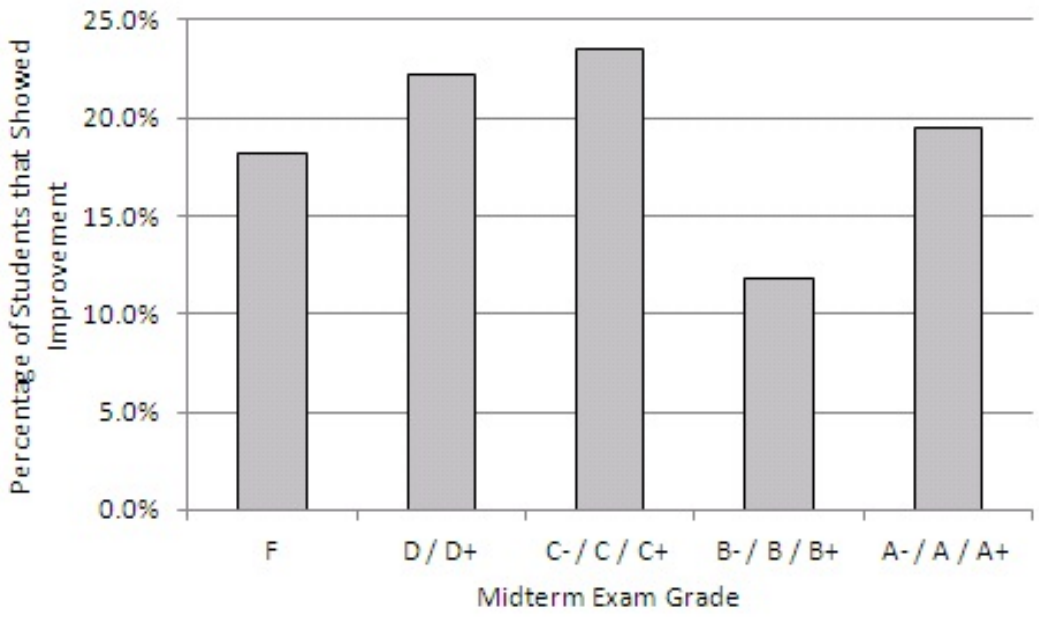


Figure 2: Distribution of Students that Improved

As Figure 2 shows, students that achieved a C or D on the midterm exam were the most likely to improve. On one hand, this is intuitive - students with these grades have among the most to gain by expending extra effort before the final exam. However, we had assumed that improvement in this category of students was less likely because their lackluster midterm exam performance indicated that they did not have a solid foundation to support learning the more advanced topics later in the course. This result suggests that the students' arguments that their midterm exam performance was not an accurate reflection of their knowledge may have some merit. It could also indicate that the midterm exam was a learning experience in addition to an assessment.

Previous work has explored the factors that make exams a learning experience [8]. In that study students wrote the same exam twice, once individually and once in a small group. This was found to improve student learning considerably. While our students didn't write the same exam twice, there are some parallels. Our students were assessed on the same material twice, in a similar way each time, and they did have ample opportunity to learn from each other, their teaching assistants or the course instructor between the two assessments.

4.5 Impact on Class Average

Before implementing this change in grading policy we had some concerns about grade inflation. Having now completed our study, we have seen that grade inflation is not a significant concern. The grade point average determined from the four assignments, original midterm exam score, and final exam score, excluding students who withdrew from the course was approximately 2.26 grade points. When the midterm exam grades were replaced, this increased to approximately 2.30 grade points, a difference of just less than 0.04 grade points. While this is an increase, it is of

sufficiently small magnitude that we do not consider it to be a problem. Furthermore, we believe that this increase is a more accurate reflection of student's knowledge at the end of the course, and as such, is fully justified.

4.6 Course Evaluations

Near the end of the course our institution conducts a survey of the students. Students are asked to rate 12 aspects of the course [6]. Overall instruction is rated on a scale from unacceptable to excellent. Students are then asked about their level of agreement with 11 additional statements such as "The course content was communicated with enthusiasm", "Students were treated respectfully" and "I learned a lot in this course". Students are also asked about their level of agreement with the statement "The evaluation methods used for determining the course grade were fair".

A Chi Square (?2) test of proportions was performed on each question to identify any differences between 2009 and 2010. The number of students that somewhat agreed, agreed or strongly agreed with positive statements were compared to the number of students that were neutral or disagreed with such statements. No statistically significant differences were observed in 11 of the 12 categories. A statistically significant decrease in the number of neutral or negative responses was observed for the statement "The evaluation methods used for determining the course grade were fair" (critical value: 3.84, observed value: 4.43). In 2009, 9.5% of students had a neutral of negative response to this question. This decreased to 2.5% of students in 2010.

5 CONCLUSION

We evaluated the impact of replacing student midterm exam grades with their final exam grades when their final exam grades exceeded their midterm exam grades. Some positive impacts were observed, including an increase in midterm exam grade for approximately 19.0% of students, and a statistically significant decrease in the number of students that were neutral or negative toward the statement "The evaluation methods used for determining the course grade were fair" on the course evaluations. However, some negative impacts were also observed. The participation rate on the midterm exam showed a statistically significant decrease from approximately 96.0% to 90.6%, and while the combined total for withdrawals and failures remained essentially unchanged, the proportion of failures increased considerably. Based on these results we believe that it would be worthwhile to continue our policy of replacing midterm exam grades with improved final exam grades. However, we also believe that additional steps should be taken to try and combat students' optimism bias so that the number of students failing the course can be reduced to 2009 levels.

REFERENCES

[1] Armor, D. A., Taylor, S. E., When Predictions Fail: The Dilemma of Unrealistic Optimism. In Gilovich, T., Griffin, D., Kahneman, D. (Eds.) Heuristics and biases: *The psychology of intuitive judgment*. Cambridge, UK: Cambridge University Press, 334 - 347, 2002.

[2] Dunning, D., Heath, C., Suls, J. M., Flawed Self-Assessment: Implications for Health, Education, and the Workplace, *Psychological Science in the Public Interest*, 5 (3), 69-106, 2004.

[3] Gibbs, G., Simpson, C., Conditions under which assessment supports student learning, *Learning and Teaching in Higher Education*, 1 (1), 3-31, 2004.

[4] Stephenson, B., Taube-Schock, C. QuickDraw: Bringing Graphics into First Year. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*, pages 211-215, 2009

[5] University of Calgary, Natural sciences program, http://www.ucalgary.ca/natsci/, 2010.

[6] University of Calgary, Universal Student Ratings of Instruction (USRI) Sample Form and Questions, http://www.ucalgary.ca/usri/ref/forms, Accessed October 12, 2011.

[7] Walker, H. M., Classroom Issues: Grading and the Allocation of Points, *SIGCSE Bulletin*, 41 (4), 14-16, 2009.

[8] Yu, B., Tsiknis, G. Allen, M., Turning exams into a learning experience, *Proceedings of the 41st ACM technical symposium on Computer science technical symposium on Computer science education (SIGCSE '10)*, 336-340, 2010.

TEACHING CAPTCHA IN A PHP PROGRAMMING COURSE*

Penn Wu and Pedro Manrique
DeVry University – Sherman Oaks
818-713-8111
{pwu, pmanrique}@devry.edu

ABSTRACT

CAPTCHA is a popular protection mechanism used by many web sites to defend against attacks from software robots which attempt to gather information or automatically fill in forms and submit them assuming someone's identity. In this paper, the authors describe how students can be taught the basic concepts of CAPTCHA programming through hands-on coding activities. The authors also discuss sample PHP code examples that can be used to illustrate the coding techniques. Instructors can use these code examples to demonstrate the various types of CAPTCHA applications.

INTRODUCTION

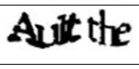
CAPTCHA is an acronym which stands for "Completely Automated Public Turing Test to Tell Computers and Humans Apart" [8]. It is a technique designed to prevent software robots (also known as "Internet bots", "web bots", or simply "bots") from interacting with a data-driven Web site [9]. "Bots" are computer programs designed to simulate human behaviors on the Web such as submitting texts, playing games, posting messages, and analyzing data. [2] describe CAPTCHA as an artificial intelligence test to prevent bots from accessing computer systems. Many web sites use CAPTCHA to determine whether the inputs such as username are manually entered by a human. A well-developed CAPTCHA application can effectively stop such attacks and minimize the unwanted commercial promotions, harassment, and vandalism [1].

According to [11] and [4], there are the four types of CAPTCHA applications:

- Text-based: Users read distorted characters that are not recognizable to current technologies of character recognition or pattern recognition. This is the most common form. Major public web sites such as Google's Gmail, Yahoo! Mail, and Microsoft Live Mail have been using this type of CATPCHA for years.

- Natural language-based: Users type in natural language phrases for validation. In Table 1, Tripod.com and Bebo.com are examples of this type of CATPCHA application. Users type two separate words in a text box for validation.
- Sound-based (or auditory CAPTCHA): Users listen to a sound clip and perform a speech recognition task. This is an alternative mechanism for people with visual disabilities.
- Image-based: Users solve an image recognition problem. For example, the yuniti.com asks users to select two matching images.

Table 1: Sample CAPTCHA Applications

Google Gmail	Yahoo! Mail	Microsoft Live Mail	Tripod.com	Bebo.com
				

Since the use of CAPTCHA as a protection mechanism against bots has become a common practice, the authors believe web programming courses, such as PHP programming, would benefit from teaching the core concept of CAPTCHA. A hands-on learning experience is one effective way to learn CAPTCHA programming and how it is integrated with web-based e-business applications. The authors, thus, share experiences and source codes with professors interesting in teaching this topic in this paper.

CAPTCHA IMPLEMENTATIONS

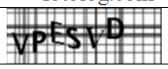
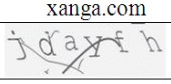
There are several CAPTCHA implementations with source code publically available on the Internet and many of them are free and non-proprietary, including Microsoft's Asirra [3]. Commonly used platforms include PHP, ASP (or ASP.NET), Perl, Python, and JSP. Table 2 provides a list of sample CAPTCHA implementations. However, they are commercial-ready tools with complicated source codes. Professors may find it difficult to explain how a CAPTCHA function is created and integrated to a Web site using these tools. In a web programming course, CAPTCHA programming could be one of many possible topics. It is the opinion of the authors that it could be easier to present this topic using a set of simple code examples as outlined in this paper.

Table 2: Sample CAPTCHA Implementations

Implementation	URL
Asirra	http://research.microsoft.com/en-us/um/redmond/projects/asirra/
Captchas	http://captchas.net/
freeCap	http://www.puremango.co.uk/2005/04/php_captcha_script_113
reCAPTCHA	http://www.captcha.net/
Securimage	http://www.phpcaptcha.org/

CAPTCHA implementations typically use an HTML form that requires users to read a distorted or scrambled set of characters, manually type in the characters, and then submit the entry for validation. To increase the complexity, many CAPTCHAs display characters on top of a dizzying texture (background image) while others add irregular lines, curves, shapes, or strange symbols for character distortion. Table 3 illustrates these CATPCHA approaches.

Table 3: Sample CAPTCHAs with dizzying textures

digg.com	fotolog.com	reddit.com	xanga.com	alibaba.com
				

SIMPLE CODE EXAMPLES

Algorithms used by commercial software are largely closely kept business secrets and may potentially remain unknown to academia. For example, Microsoft attempts to patent its CAPTCHA technologies. In keeping with interests outlined, the authors have developed simple code examples in order to teach the core concepts of a CATPCHA application.

Text-based CAPTCHA Applications

The following is a complete PHP code that generates a PNG image. The image will display five randomly selected characters. The font size of these characters is randomly generated. Each character is also rotated to a randomly selected degree. Notice the functions: imagecreate, imagecolorallocate, imagefttext, and imagedestroy which are functions provided by the Graphics Draw (GD) library for PHP.

```
01 <?php //Filename: captcha.php
02 $str = ""; // a null varaible
03 $captcha = "";
04 $img = imagecreate(180,70); // specify the image size
05 $bg = imagecolorallocate($img,255,255,255);
06 $font_file = "arialbd.ttf"; // specify the path to the font file
07 for ($i=0; $i<5; $i++) {
08  switch (rand(0,2)) {
09   case 0: $str .= chr(rand(65,90)); break; // randomly select an
uppercase letter
10   case 1: $str = chr(rand(97,122)); break; // randomly select a
lowercase letter
11   case 2: $str = chr(rand(48,57)); break; // randomly select a
numeral
12  }
13  $captcha .= $str; // concatenate the characters
14  $font_size=rand(15,35);
15  $x += $font_size;
16  $y=rand(0,10)+45;
17  $char_color =
imagecolorallocate($img,rand(20,255),rand(20,255),rand(20,255));
18  imagefttext($img, $font_size, rand(-30,30), $x, $y, $char_color,
$font_file, $str);
19 }
20 ob_start();
21 imagepng($img);
```

```
22 $imagevariable = ob_get_contents();
23 ob_end_clean();
24 $base64_data = base64_encode($imagevariable);
25 imagedestroy($img);
26 ?>
27 <img src='<?php echo $base64_data ?>' />
```

Line 7 to 12 is a for loop that uses PHP's rand function to generate random integers in the range of 0 to 1. It also uses the chr function to return ASCII characters. The range, 65 to 90, is ASCII codes for uppercase letters, while 97 to 122 are lowercase letters and 48 to 57 are numerals. An alternative is to adopt Kumar's algorithm [6], in which the PHP microtime and md5 functions are used to generate the random string, and then trim the long string by the substr function as shown below.

```
$RandomStr = md5(microtime());
$ResultStr = substr($RandomStr,0,5);
```

One known problem is that that the GD library is only responsible for generating data to create images (through functions like imagepng). As a result, developers need to use proper PHP functions to save image data to a memory buffer (or variable). In the above code, the authors decided to use the following PHP functions.

- ob_start() -Turn on output buffering
- ob_end_clean() - Clean (erase) the output buffer and turn off output buffering
- ob_get_contents() - Return the contents of the output buffer

PHP is a server-side language in that it generates and sends the image data to client browser for execution. The authors use the PHP base64_encode function to encode image data with the Base64 encoding of MIME (Multipurpose Internet Mail Extensions) standard. The image data will be converted to a series of characters as shown below. It must be noted that Base64-encoded data shall increase the data size by approximately 33%.

`iVBORw0SNk09CZDDihhKbkK1mS7f3j2MfH93HuuQ85m.....` (Image data encoded as base64)

The above encoded string may be incorporated to a URI (Uniform Resource Identifier). URI is used to identifying resources in a TCP/IP network. An image generated by PHP is an object on the Web and its image data is the resource to be retrieved by the client browser. A URI can be accessed by a URL (Uniform Resource Locator). A URL is a specialization of URI that defines the network location of a specific representation for a given resource. According to the RFC 2397 "data" URI scheme [7], URLs that define data source must begin with protocol type "data:" rather than "http:" The format of Data URI scheme is:

```
data:[<MIME-type>][;charset=<encoding>],<data>
```

The captcha.php file generates PNG images; therefore, the MIME-type is image/png. Since the image data is stored in the $base64_data variable and is encoded as base64, the URI in Line 27 will serve as an HTTP URL for the src attribute of an IMG tag. The src attribute specifies the URL of an image. Line 27 is a statement for reading the encoded image data and displaying the image on the client browser. Consequently the captcha.php file creates a 180×70 PNG image as specified in Line 4.

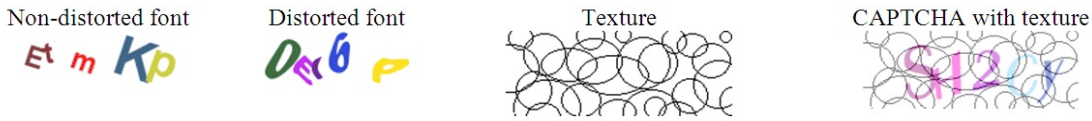


Figure 2: Texture File Examples

The captcha.php file can be accessed directly referencing the URL. For example, if the captcha.php is uploaded to a web server with an address http://www.cypresscollege.edu, then the HTML code to invoke it can simply written as:

```
<img src="http://www.cypresscollege.edu/captcha.php">
```

The GD Library does not support any functions that can distort an image. Moreover, the distortion algorithm is computationally intensive and PHP as a server scripting language is not designed for intense computing tasks. An easy alternative is to use distorted fonts. They are available for downloading at the http://www.dafont.com/ site. To use distorted fonts, simply upload the font file to the web server and precisely point to the path of the font file as illustrated by the following line of code.

To increase the complexity, students can prepare a texture file similar (see Figure 2 above) and then use it to create CAPTCHA images with dizzying texture. The GD library has many image functions such as imagecreatefrompng for copying image files. The following statements copy the texture file (bg.png). The imagecopymerge function will merge the texture with five characters. Figure 2 illustrates the result of merging textures and various font styles.

```
$img2 = imagecreatefrompng('bg.png');
imagecopymerge($img2, $img, 0, 0, 0, 0, imagesx($img), imagesx($img),
50);
```

In Line 13, the $captcha variable concatenates (or combines) the five characters to a string. The following statements create a PHP session variable named "key" which can store these five characters for later use. According to TCP/IP, a session (or link) is created between client and the remote server for passing data each time a client connect to the server. The session is destroyed as the user leaves (or disconnects) the server.

```
session_start();
$_SESSION['key'] = $captcha;
```

The following statements create an HTML form for users to submit inputs. $_SERVER['PHP_SELF'] is an PHP "Predefined Variables" that returns the filename of the currently executing script. It instructs the "action" attribute to send the user inputs to the captcha.php file (at the server side) for validation. Figure 3 provides a sample result of the following HTML form.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="Text" name="captchas">
<input type="submit" value=" Go ">
</form>
```

The following *if* statement validates the user inputs. $_POST['captchas'] retrieves the user inputs (entered to the "captchas" textbox) and passes the value to the $captchas

variable. The *if* statement determines if the user inputs match to the value stored in the session variable $_SESSION['key'].

```
If ($_POST){
 session_start();
 $captchas = $_POST['captchas'];
 if($captchas == $_SESSION['key'])
  { echo "Welcome!"; }
 else { echo "Invalid! Try again!";}
 exit;}
```

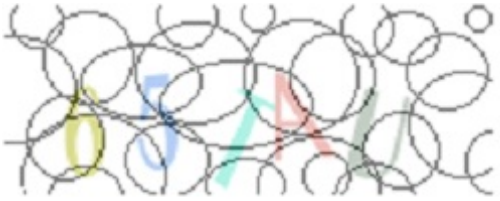


Figure 3: A completed HTML form of CAPTCHA

Image-based CAPTCHAs

Image-based CAPTCHAs are less common. [10] proposes an algorithm to slightly distort a small portion of an image. A human can easily find the distortion and click on the distorted area to pass the CAPTCHA test. With the imap.php file the authors create a simple implementation of image-based CAPTCHAs. The authors' algorithm simply places a colored-filled, half-transparent square on a randomly selected location of a PNG image file as shown in Figure 3. Human users can easily find the square and click on it to pass the CHAPTCHA test, but the state-of-the-art technology of "bots" cannot.

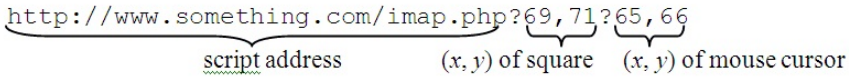
```
$rect = imagecreate(10,10);
$bg   =   imagecolorallocate($rect,
255,255,0);
imagerectangle($rect, 0, 0, 10, 10,
$bg);
$img = imagecreatefrompng('cg.png');
.................
list($width, $height, $type, $attr)
= getimagesize("cg.png");

$rectX = rand(0, $width-10);
$rectY = rand(0, $height-10);

imagecopymerge($img, $rect, $rectX,
$rectY,   0,   0,   imagesx($rect),
imagesy($rect), 50);
```

Figure 3: Image with a yellow square

The above PHP statements create a yellow square and merge it with a photo image (named "cg.png"). The width and height of square are 10 pixels. The *getimagesize* function of GD will return width and height of the image. By enclosing the square inside the photo image, the PHP rand function will randomly generate value of x- and y-coordinates of the square with respect to x- and y- coordinates of the photo image. Consequently, the location of square is not foreseeable.

The imap.php file uses the ISMAP attribute of IMG tag to specify that the image uses a server-side image map. When clicking any pixel of the image file, the x- and y-coordinates of the mouse cursor will be posted to the URI as illustrated below. The

question mark (?) is delimiter. The authors add two session variables, $_SESSION['x'] and $_SESSION['y'], to attach the x- and y- coordinate of the square.



The PHP *preg_split* function will split the URI to an array named *"str"* with three elements: script address, (x, y) of square, and (x, y) of mouse cursor. The *preg_split* function will then split the value of $str[1] to an array named "str1" which has two elements: str1[0] stores x- coordinate and and str1[1] stores y-coordinate of square. The preg_split function also split the value of $str[2] to an array named "str2". $str2[0] holds the x-coordinate and $str2[1] holds the y-coordinate of mouse cursor.

```
session_start();
$str=$_SERVER['REQUEST_URI'];
$str=preg_split('/\?/', $str);
$str1=preg_split('/\,/',
$str[1]);//square x,y

$str2=preg_split('/\,/',
$str[2]);//cursor x,y
if ($str2[0] >= $str1[0] &&
    $str2[0] <= $str1[0]+10 &&
    $str2[1] >= $str1[1] &&
```

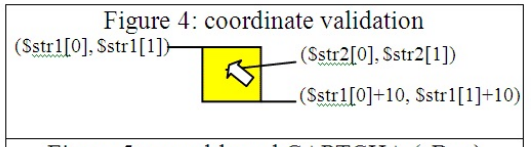


Figure 4: coordinate validation

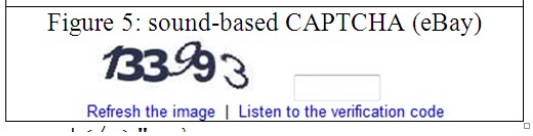


Figure 5: sound-based CAPTCHA (eBay)

The *if* statement can then determine whether the user clicks a pixel inside the yellow square. Only when the cursor's x-coordinate is between square's x-coordinate and square's x-coordinate plus 10 (which is the width of square) and cursors' y-coordinate is between square's y-coordinate and square 's y-coordinate plus 10 (which is the height of square), the cursor's pointer is verified to be inside the area of the square (see Figure 4).

Sound-Based CAPTCHAs

Both text-based and image-based CAPTCHAs can present problems for color-blind users [5]. They can also significantly increase the accessibility issues to those who have vision issues or suffer from a cognitive disability such as dyslexia. Using sound-based CAPTCHA is an alternative. For example, eBay.com's CAPTCHA provides a "Listen to the verification code" option as shown in Figure 5.

```
02 <?php //Filename: audio.php
03 session_start(); $str = ""; $captcha = "";
05 for ($i; $i<5; $i++) {
06     $r   =   ord(substr(session_id(),   $i,
$i+1))%9;
07  $captcha .= $r;
08  $str .= "http://www.cypresscollege.edu/"
. $r . ".wma\n"; }
09 $_SESSION['key'] = $captcha;
10 header("Content-Type: audio/mpegurl");
11 header("Content-Disposition: inline;
filename=captcha.m3u");
12 echo $str; ?>
```

Figure 6: Auditory CHAPTCHA

The audio.php file is the authors' implementation of sound-based CAPTCHAs. It will generate a M3U playlist file named "captcha.m3u" to read out the CAPTCHA characters, as specified in the "filename" header field of MIME protocol. The .m3u file extension stands for "MP3 URL" or "Moving Picture Experts Group Audio Layer 3 Uniform Resource Locator". M3U is a format created by the Winamp media player to store multimedia playlists and it is now widely supported by several multimedia applications. A M3U file is a plain text file with the ".m3u" extension. For sake of completion, the authors recorded their sound-based CATPCHAs as individual MP3 files.

The value "inline" of the "Content-Disposition" field forces the client browser to play the contents of the playlist file. As shown in Figure 6, users can click the "Play Sound" hyperlink to listen to the pre-recorded sounds. This hyperlink simply calls the audio.php file to as shown below.

```
<a href="audio.php">Play Sound</a>
```

The captcha.m3u file contains a list of five URLs. Each URL points to a pre-recorded sound file. Each pre-recorded file play a sound to identify an individual CAPTCHA character such as "One", "two", and so on. If the randomly generated CAPTCHA string is "84703" (as in Figure 5), the captcha.m3u file will contain only the URLs to five sound files:

http:://www.domain_name.com/sounds/8.wma
http:://www.domain_name.com/sounds/4.wma
http:://www.domain_name.com/sounds/7.wma
http:://www.domain_name.com/sounds/0.wma
http:://www.domain_name.com/sounds/3.wma.

They will be played by the client browser in the order that they are listed in the file.

CONCLUSION

CAPTCHA has been developed over 10 years ago and is now utilized by many website designers to provide an additional security measure to protect their website and users from potential harm. The authors believe CAPTCHA should be one of several critical skills amenable to being integrated at career-oriented schools.

Career-oriented schools have the responsibility to deliver the most current and essential job skills that are readily transferable to the industry and can facilitate the students' entry into the workforce. This paper demonstrates how to present and/or augment a course curriculum to teach a new topic, such as CAPCHA. In this case, PHP is one of the most popular server side scripting languages and is becoming the dominant development platform for data-drive web sites; therefore, it makes sense to add a section on CAPTCHA to PHP course material. With the support of GD Library, PHP is an ideal language for developers to create simple CAPTCHA applications. Professors can readily integrate this topic within regular PHP programming courses by adding the discussion of the core concepts of CAPTCHAs using the PHP code examples presented in this paper.

REFERENCES

[1] Arora, A. (2007). Statistics hacking: Exploiting vulnerabilities in news websites. *International Journal of Computer Science and Network Security,* 7(3), 342–347.

[2] Caine, A., & Hengartner, U. (2007). *The AI hardness of CAPTCHAs does not imply Robust Network Security*. http://www.cs.uwaterloo.ca/~uhengart/publications/ifiptm07.pdf, retrieved September 7, 2011

[3] Dignan, L. (2007). *Microsoft Windows Live Mail's CAPTCHA defense falls to spam bots*. http://www.zdnet.com/blog/security/microsoft-windows-live-mails-captcha-defense-falls-to-spam-bots/863, retrieved October 23, 2011

[4] Godfrey, P. (2002). *Text-based CAPTCHA algorithms*. http://www.aladdin.cs.cmu.edu/hips/events/abs/godfreyb_abstract.pdf, retrieved October 23, 2011

[5] Holman, J., Lazar, J., Feng, J. H., and D'Arcy, J. (2007). Developing usable CAPTCHAs for blind users. *Proceedings of the 9th international ACM SIGACCESS Conference on Computers and Accessibility*, 245-246.

[6] Kumar, S. (2006). *Securing Web forms with simple PHP-CAPTCHA*. http://www.codewalkers.com/c/a/GUI-Code/Simple-PHPCAPTCHA/, retrieved September 7, 2011

[7] Masinter, L. (1998). *The "data" URL scheme. Network Working Group*. http://www.ietf.org/rfc/rfc2397.txt, retrieved October 23, 2011

[8] May, M. (2005). *Inaccessibility of CAPTCHA: Alternatives to visual Turing Tests on the Web*. http://www.w3.org/TR/turingtest/, retrieved October 3, 2011

[9] Mori, G., & Malik, J. (2010). *Breaking a visual CAPTCHA*. http://www.cs.sfu.ca/~mori/research/gimpy/, retrieved September 7, 2011

[10] Tarasyuk, M. (2007). *Image-Based CAPTCHA*. http://marss.co.ua/ImageBasedCAPTCHA.aspx, retrieved October 23, 2011

[11] Yan, J. & El Ahmad, A. (2008). Usability of CAPTCHAs or usability issues in CAPTCHA design. *Proceedings of the 4th Symposium on Usable Privacy and Security*, Vol. 337, 44-52.

USING SCRUM IN A QUARTER-LENGTH UNDERGRADUATE SOFTWARE ENGINEERING COURSE*

Linda Werner, Dominic Arcamone, Ben Ross
University of California
Santa Cruz, CA 95064
831 459-1017
linda@soe.ucsc.edu, darcamon@ucsc.edu, bpross@ucsc.edu

ABSTRACT

In this paper, we describe the use of Scrum in an upper-division software engineering quarter-length course. We describe tools used to support both the use of Scrum and the overall objectives of an introductory course in software engineering. We do this to provide support for others who want to teach an introductory software engineering course in a format suitable for a course shorter than the typical semester-length course.

INTRODUCTION

Many computer science departments offer upper-division courses in software engineering. Typically, one goal of these courses is to impart the theory and practice of software engineering by means of student groups creating a software system. Students form teams, use one of the software development models as described in the Software Engineering Body of Knowledge and create a software system chosen either from a list provided by the instructor or from those suggested by the student teams themselves but requiring instructor approval.

This paper focuses on the benefits of using Scrum in these kinds of courses by describing the overwhelming benefits seen in the first use of Scrum for a course that has been taught more than twenty times since 1985. The course description written more than two decades ago states the course emphasizes the characteristics of well-engineered software systems. Topics include requirements analysis and specification, design, programming, verification and validation, maintenance, and project management.

Practical and research methods are studied. The course imparts an understanding of the steps used to develop computer software. Additionally, the course satisfies the writing-in-the-major requirement for computer science students at this university, so there are numerous writing assignments and feedback regarding syntax and content is given to the students.

Scrum is an Agile Project Management Methodology using an iterative and incremental software development method. Agile methods are considered to be "adaptive rather than predictive" and, therefore thrive on change; and "people-oriented rather than process-oriented" suggesting that software development should be an enjoyable process [3]. Scrum has been successfully used for education [7], management of projects in graduate courses [5], has seen adoption in semester-long software engineering and game development university courses [4][9], and has been used for three years in the three-quarter, year-long game design studio program at this university.

The format of the rest of this paper is as follows: first, an outline of the software engineering course is given. Each week's class activities, deliverables, and lecture topics are listed. Activities are described with sufficient detail so that others may adapt this outline for their courses. Next, the Three-sprint Scrum, the modified Scrum used in the software engineering course, is compared to industrial Scrum highlighting the differences and pedagogic reasons for these changes. Finally, problems with the use of Three-sprint Scrum are discussed.

COURSE OUTLINE

This software engineering course is 10 weeks long with 2- 1 3/4 hour classes and 1 lab hour per week. The students are junior and senior computer science, game design, computer engineering, technology and information systems, and bioinformatics majors and are encouraged to spend approximately 15 hours per week on activities associated with this course.

TABLE: Software Engineering Course Contents

Week	Lecture topics	Activities	Deliverables
1	1. Introduction to software engineering 2. Introduction to Scrum	1.Watch Agile in Practice: Frequent Small Releases, Story Cards/User Stories, Prioritisation using MoSCoW, Planning Poker 2.ice breaker exercise (see below) 3. Planning poker in-class exercise using a high priority user story from each project team 4. Read-and-Comment (RAC) "The New Product Development Game", by Takeuchi and Nonaka	1.Determine team members 2.Registration of team for Gforge (using Subversion) 3.Registration of team for Scrum tool: PivotalTracker
2	1.Software development process models 2.Software development workflows 3.More Scrum 4.Version Control	1. Begin sprint 1 2. Watch Agile in Practice: Burn-up charts 3.quiz on Scrum terminology 4. RAC "Version Control Systems" by Spinillis and "Making Sense of Revision-control Systems" by O'Sullivan 5.Class presentations of teams' planned software systems	1.Team Status report (TSR) 2.Release plan 3.Class presentation of team & software system

3	1.UML 2.XP 3.Agile processes 4.CMM	1.RAC "Extreme Programming from a CMM Perspective" by Paulk 2.Position Scrum into the KPA's of CMM done in small groups as an in-class exercise. One group's solution presented 3.UML class diagram problems done in small groups. Students' solutions are presented	1.Sprint 1 plan 2.TSR 3.updated burndown chart and Scrum board
4	1.Software engineering team structures 2.software inspections using SEI video	1.Class presentations of teams' technical designs 2.Begin sprint 2	1.Technical design with UML class diagrams and sequence diagrams 2.TSR 3.Sprint 1 report 4.Updated sprint burndown chart and scrum board 5.Sprint 2 plan and updated release plan if necessary
5	1.non-execution based testing 2.implementation workflow	1.watch Agile in Practice: Burn-up charts/Definition of Done 2.quiz on process models, CMM, testing, and UML	1.TSR 2.Updated sprint burndown chart and Scrum board
6	1.execution based testing	1.each team does inspection (one done in class, others in lab) 2.Begin sprint 3	1.Sprint 2 report 2.TSR

7	1.information hiding 2.cohesion and coupling	1.cohesion and coupling exercises done in small groups 2.RAC "What is Software Testing? And Why is it So Hard?" By Whittaker	1.Sprint 3 plan and updated release plan if necessary 2.TSR
8	1.requirements engineering 2.pair programming 3.industrial swdevelopment	1.RAC "How Pair Programming Really Works" by Wray 2.guest speaker from industry	1 TSR 2.Updated sprint burndown chart and Scrum board
9	1.integration 2.CASE tools	1.quiz on testing, cohesion and coupling, implementation, integration 2.peer review of user documentation	1.user documentation 2.TSR 3.Updated sprint burndown chart and Scrum board
10	1.acceptance testing	1.acceptance testing of all team projects	1.TSR 2.final project submission including sprint 3 report 3.project reflection essay

The course is built on the application of the reflective practitioner perspective as applied to software engineering education [2]. Hazzen builds on work by Schon [10] and argues that in order to teach about approaches and methodologies for the development of software, student's understanding of these "should be based on one's personal experience and reflection on one's creation process." Additionally, Hazzen writes, "in the context of SE (software engineering), the better one understands the process of developing a software system, the better one may understand the methodologies that guide this process." We believe this to be the case, and teach the principles of software engineering by guiding students through the process of developing software by use of a modification of industrial Scrum, that we call Three-Sprint Scrum, and using a software development process model (iterative and incremental), asking students to periodically reflect on the use of those processes, and participate in other activities in which other software development process models are discussed.

The first class of the quarter starts with an introduction to software engineering. The classic waterfall software development process model is discussed describing the workflows of requirements, analysis, design, implementation, and test; followed immediately by an introduction to industrial Scrum using short videos on industrial Scrum practices available on the Internet [2]. The course syllabus is discussed,

milestones for using Three-Sprint Scrum to manage the creation of a software system are identified, and deliverables are described. At the end of this first class session, students work in groups of five or six on a set of four questions. The teams choose a note-taker, a moderator, and a timekeeper and are given approximately 15 minutes to answer the questions. The answers are hand-written and submitted at the end of class. The questions are:

- What are the names, emails, majors, and years of each student?
- What are skills, work experience, and previous courses believed to assist in this course?
- What are each student's skills, experience, and courses from the list created in the previous answer?
- What skills and previous courses are missing in the group?

This exercise has multi-purposes. This is a way for students to discuss the lecture material. They have to verbalize what they understand to be important to be successful in this software engineering course. This introductory exercise serves as an icebreaker. Working with classmates means that everyone meets at least four other students enrolled in the course. Students get to exchange names, emails, and leave the classroom discussing potential project ideas.

By the end of the first week of classes, students have been given class time to choose their project teams, discuss potential project ideas, and experiment with Planning Poker. Planning Poker is an industrial Scrum practice in which team members estimate user story development time using a consensus, Delphi-like process. During the first week of classes, students use class time to experiment with Planning Poker to estimate the time required for one of their highest priority user stories. They are encouraged to continue to use Planning Poker to estimate the development time needed for all of their project's user stories. Potential projects have included computerized games, SCORE projects [11], textbook projects [8], software systems needed by university professors in their research projects, and projects suggested by students.

Most classes are structured with a lecture component and short exercises where students are presented with problems that are discussed in small groups of two or three students. These groups are always comprised of project team members (never cross-team) in an attempt to encourage team jelling. One or more volunteers from these small groups are given the opportunity to present their solutions to the rest of the class. The instructor freely asks questions during these presentations and students are encouraged to join in the discussions. The classroom is equipped with an Internet connection, computer with large-screen display capabilities, a document camera, and flash drive access. Students and student groups get ample opportunities throughout the quarter to present their work to the class using the various technological capabilities of the classroom. These in-class activities, as well as formally scheduled presentations, give students practice with technical presentations, a necessary component of a software engineering course and one that satisfies a university-wide requirement for upper-division disciplinary-based writing. This requirement calls for various writing assignments specific to the discipline of computer science with feedback on content and style provided to students. Course assignments include individually written comments on readings of five journal papers on important software engineering topics (called

read-and-comment or RAC in the above table), various team plans and reports, a peer-review exercise, and various formal presentations (team and software system plan, software technical design, and final system acceptance test). Additionally, for at least one assignment, the in-class peer-review, the students have an opportunity to revise the writing based on feedback.

THE THREE-SPRINT SCRUM

Student project teams manage the development of their systems using a modified form of industrial Scrum that we call the Three-Sprint Scrum. Industrial Scrum is an agile project management methodology in which teams are composed of software engineers and a representative of the user, called the Product Owner. Another person, in the role of the Scrum Master, facilitates industrial Scrum teams' activities and oversees that the proper industrial Scrum process is followed. A release of the software is produced by means of a number of sprints, each focused on the set of user stories (an increment) and in the order deemed important to first create an executable version of the product and then to extend the product's functionality until a fully functioning product has been created and release. For industrial Scrum teams, one Scrum Master may participate in that role for many industrial Scrum teams.

For our course, both the role of the Product Owner and the Scrum Master are filled by fully participating team members. Our student teams' Product Owners hold that position during the entire course in addition to completing other project tasks. They are the liaison between the instructor and any external project owner and the team. If the product idea is one envisioned by the team, then the Product Owner has the final word on user story specifics and prioritization issues. The student team's Scrum Master is determined at the start of each sprint. Our students have 3 sprints, so a new Scrum Master is chosen 3 times. All students participate as fully functioning team members and also as either the Scrum Master for a sprint or as the Product Owner for the length of the release. Since our teams have 5-7 members, a sprint may have 1 or 2 Scrum Masters. A Product Owner may never become a Scrum Master. The structure of the project teams gives each team member an opportunity to manage a small part of the project.

Activity or characteristic	**Industrial Scrum**	**Three-Sprint Scrum**
Size of team	5-9	5-7
Length of sprint	2-4 weeks	2-4 weeks
Participant work week	40 hours	8 hours
Product Owner	Outside Scrum team	Fully participating member of Scrum team; static role throughout course

Product backlog determined by	Product Owner	Students with input from instructor during a product-planning meeting; ties are broken by one student per team assuming the Product Owner role. The meeting culminates in a release plan (written deliverable) that lists the items in the product backlog
Sprint backlog determined by	Scrum Team	Scrum Team
Sprint plan	Sprint planning meeting	Sprint planning meeting culminating in sprint plan (written deliverable)
Sprint review and retrospective	Two meetings	Meeting resulting in a sprint report (written deliverable)
Scrum Master	Outside Scrum team	Fully participating member of Scrum team; 2 members may share role; role rotates throughout Scrum team with all team members participating as a Scrum Master except Product Owner; new Scrum Master(s) are chosen at the start of each sprint
Scrum meeting	Daily	3 times a week; one of those meetings scheduled with instructor, tutor, or TA. One team status report per person per week is a written report explaining team member's impression of the contribution of each other team member.

We've instituted additional written reports (Release plan, three Sprint plans, three Sprint reports) that all provide for additionally visibility into the progress of the project development process and are used for grading purposes. Without these reports, the instructor, tutor, or TA would have to be present at every team meeting. Additionally, these reports are used to satisfy the writing-in-the-major requirement.

Industrial Scrum dictates the use of a publicly visible burndown chart and Scrum board. Because a dedicated room is not available for this, we use an educational use of Pivotal Tracker [6], an online CASE tool for use with industrial Scrum and other agile process models. Our university has one Pivotal Tracker account and we are able to create an unlimited number of projects, one for each team. All TA's, tutors, and the instructor are members of all project teams. We also use a special software tool, the Team Status

Reporting system, created by a graduate student for use by the Game Design Studio and modified for use in the software engineering course. All team members report weekly on their own and their team members' accomplishments according to what they had agreed to when the sprint started. In addition, they report fellow team members' activities they especially like and those they don't like. On the Team Status Report weekly due date, each team member can view reports written about themselves labeled with the author's name. These reports are also visible to the instructor, TA, and tutors as soon as they are submitted. These reports help everyone: the instructor, TA's, and tutors use this to determine if teams are jelling and to determine if intervention is required to assist the team in interpersonal communications and management. The members of the teams use it to learn whether their impression of their activities matches what others on their teams believe. It improves team communication and helps team members know when their teammates need help to accomplish their own tasks. Without these team-building activities, groups may encounter problems with individual team members who are not accustomed to working in groups. Lower-division game design majors have required team projects, however, lower-division computer science courses recommend, but don't require students to work in pairs and groups.

STUDENTS' COMMENTS

The last deliverable is the 'Project Reflection Essay.' The following quotes are from the essays and are representative of the comments I received this first course offering using Scrum:

"Overall, the start-to-finish process of inventing, developing, and testing … went surprisingly well."

"Most of my programming work sessions ended up being done with pair programming, which was really helpful. As for communication between people in different locations, the best interactions came during the Scrum meetings, where we constantly re-evaluated tasks and kept each other updated on our performance. Besides that, we talked primarily through Gmail, either with actual emails or with Gmail chat. Those emails were usually just for clarification of topics discussed at the Scrum meetings, but sometimes significant new information was passed out over email, such as tasks being divided in a new way or a major "road block" in one or more tasks."

"Every time I turned around there was another document due. …it was getting in the way, but that was only because I was not used to…planning. After a couple of weeks …Scrum…became second nature."

I think the priority system in Scrum really put in perspective what were the most important tasks... Every time I worked on a project in my previous classes I almost always ran out of time and could not put everything … I wanted. I had a problem with over scoping ... With Scrum I learned to prioritize tasks with my team so that when time did run out at least the system critical components were implemented."

"This whole quarter has been a transforming quarter for me."

"I'm moving through my own capability maturity model. Heroic programming is … not sustainable."

"I am glad to say that it was a wholly positive experience… That's not to say that no mistakes were made, or that there weren't setbacks, but every mistake was overcome and learned from, and I feel that I am better prepared for the next group software project I work on."

"… our experience with Scrum, I think that our group was largely happy with it, although we were hindered by the tools we used, namely Pivotal Tracker. … Besides Pivotal Tracker, our use of tools was very effective. Gforge/Subversion were very helpful in keeping track of the group's development and isolating faults. We also used Google Docs for collaborating on the assignments to allow everyone to have a chance to add their input to sprint plans, presentations, etc."

PROBLEMS WITH THREE-SPRINT SCRUM AND PLANS FOR CHANGE

Many different software development process models have been used in the teaching of this software engineering course since 1985; however, no formal project management methodology has been taught and used. Sometime graduate students in a graduate software engineering course have acted as managers of the undergraduate student teams but mostly the instructor, TA's, and tutors have managed the teams. Two common managerial problems have been alleviated with the use of Three-Sprint Scrum and the added tools (Team Status Report system and Pivotal Tracker). Often it has been difficult to be aware of the progress of the teams. Using Three-Sprint Scrum has increased the visibility into the teams' progress. With the use of the Team Status Report system, the instructor, TA's, and tutors keep their fingers on the pulse of the team members' interactions. We are still learning about the best way to use Pivotal Tracker in this course setting. The students have expressed the desire for the use of a physical Scrum board with a dedicated room because they were not able to be effective with Pivotal Tracker in such a short period of time. Students in the Game Design Studio have a dedicated room and the students in the software engineering course seem to be envious of the space. Plans include further experimentation with Pivotal Tracker to determine if some of the written reports can be replaced with features of Pivotal Tracker. Alternatively, we will investigate the cost of creating a system of lockable Scrum boards that can be mounted on the walls of a public space within the department. Each team can be provided with keys to their team's Scrum board.

ACKNOWLEDGMENTS

Thanks go to Jim Whitehead, for suggesting the use of Scrum in this course. Thanks go to professors Michael Mataes and Jim Whitehead, whose use of Scrum in the Game Design Studio provided some of the lecture materials. Thanks go to Brandon Tearse, who wrote the Team Status Reporting tool.

REFERENCES

[1] ENNOVA, Agile in Practice Videos, 2011. http://ennova.com.au/blog/2011/05/agile-academy-videos, retrieved October 26, 2011.

[2] Hazzan, O., 2002, The reflective practitioner perspective in software engineering education, *Journal of Systems and Software*, 63, (3), 2002, 161-171.

[3] Hazzan, O.; Dubinsky, Y., 2003.Teaching a software development methodology: the case of extreme programming, *Software Engineering Education and Training*, (CSEE&T 2003). Proceedings. 16th Conference 176- 184, 20-22.

[4] Mahnic, V., 2011. A capstone course on agile software development using Scrum, IEEE Transactions on Education, 54(2), 273-278.

[5] Pinto, L.; Rosa, R.; Pacheco, C.; Xavier, C.; Barreto, R.; Lucena, V.; Caxias, M.; Figueiredo, C.M., 2009. On the use of SCRUM for the management of practical projects in graduate courses, *Frontiers in Education Conference*, FIE '09. 39th IEEE ,.1-6, 18-21.

[6] Pivotal Labs, Inc., http://www.pivotaltracker.com/?gclid=CNykq_6SiKwCFQSDhwodhHP8YA, retrieved on October 26, 2011.

[7] Rico, D. F. and Sayani, H.H., 2009. Use of agile methods in software engineering education, AGILE Conference, 174-179.

[8] Schach, S., 2008. *Object Oriented Software Engineering*, McGraw-Hill Textbooks.

[9] Schild, J., Walter, R., and Masuch, M., 2010. ABC-Sprints: adapting Scrum to academic game development courses, *Proceedings of the Fifth International Conference on the Foundations of Digital Games (FDG '10)*. ACM, New York, NY, USA, 187-194.

[10] Schon, D.A., 1983. *The Reflective Practitioner*. Basic Books, Inc.

[11] SCORE - Student Contest on Software Engineering, http://score-contest.org/2011/index.php, retrieved on October 26, 2011.

LEARNING FROM DATABASE PERFORMANCE BENCHMARKS*

Jennifer Ortiz, Suzanne W. Dietrich, Mahesh B. Chaudhari
Mathematical and Natural Sciences
Arizona State University
Phoenix, AZ 85069-7100
602-543-5628
{jennifer.ortiz, dietrich, mahesh.chaudhari}@asu.edu

ABSTRACT

This paper illustrates an approach to incorporate performance benchmarks in a database course to introduce students to scalable test data and query performance. The realistic TPC-H benchmark enterprise forms the basis of class exercises and homework assignments that reinforce SQL query reading and writing skills. The integration of the benchmark supports the fundamental learning objectives of the course and exposes students to additional concepts used in industry.

INTRODUCTION

Throughout the advances of the database field, instructors continually reconsider which topics to include in their introductory database course [18]. Although some may have the opportunity to offer a second database course for undergraduates [5], this option is limited. Given the recent surge of new technologies, handling this constraint has proven to be an ongoing challenge for database educators [1].

A consequence of this dilemma may result in undergraduate students lacking exposure to real-world domains and applications of databases in industry [16]. To aid understanding, students are typically exposed to simpler examples, which may lead to a "small database mindset" [6]. This results in a learning experience that may disengage students from the need for performance modifications that may be required in a larger database [6]. Performance benchmarks are actually encouraged in the expansion of the database curriculum [2, 9, 10, 12]. However, integrating benchmarks in an introductory

---

database course is not a simple process. Some of the challenges instructors may face include the need for providing a large database, a realistic concurrent environment, and even supplying "identical" hardware for each student [2].

This paper illustrates an approach to incorporate an introduction to performance benchmarks in a database course. The overall goal is to introduce students to scalable test data and query performance. This approach uses the TPC-H [14] decision support benchmark as a sample enterprise for class exercises and homework assignments. TPC-H is based on a business model of customer-orders and includes a set of 22 decision support queries written in SQL. The TPC-H specification is nontrivial and illustrates the importance of setting up an appropriate environment for testing. The queries based on the TPC-H enterprise provide real-world examples for students to learn advanced query constructs while reinforcing critical query reading skills. In addition, using the TPC-H enterprise for a homework assignment provides a realistic database scenario for writing queries and discussing the performance of alternative query formulations.

OVERVIEW OF THE TPC-H BENCHMARK

TPC-H is a decision support benchmark that is based on a realistic application involving customers, parts, lineitems, suppliers, and orders. The benchmark is composed of a database holding large volumes of data along with a set of complex queries that answer critical business questions. A program called DBGEN, provided by TPC-H, produces data for the enterprise at a scale specified by the user. This program generates data from a minimum scale of 1 GB, with fixed scale factors up to 100 TB. TPC-H provides a performance metric known as the TPC-H Composite Query-per-Hour Performance Metric that depicts the system's capability to process queries. This metric can measure the ability to manage queries through a single stream or even determine the throughput of the queries when submitted by concurrent users [14].

Since students typically lack the permission to bulk insert large amounts of data into their school accounts, this work modified the DBGEN code to scale down the database to 1 MB while satisfying the constraints given in the benchmark specification. Despite the low scale factor, parameter values have been determined so that all 22 queries generate a nonempty result in the reduced data set. These modified queries along with the SQL data definition scripts for the 1 MB TPC-H database instance are available on the author's web site [4] for various database products.

Figure 1 displays an overview of the TPC-H schema. Each box represents the name of a table. Each directed edge indicates a primary-foreign key relationship. The numbers above each table represent the number of tuples at a specific scale factor. The numbers held between brackets represent the amount of tuples at a 1 MB scale factor, while the numbers outside the brackets represent the number of tuples at the default scale factor of 1 GB.

TPC-H includes a set of 22 ad-hoc business-oriented queries that include a variety of operators and selectivity constraints [14]. Table 1 provides a characterization of each query in the benchmark. A typical benchmark query uses an inner join, although there is a query that illustrates an outer join. In most cases, the queries utilize basic aggregate functions and include a group by clause. The terms simple or composite in the Group By column indicate whether the grouping is over a single attribute or multiple attributes,

respectively. There are some queries that use a having clause on the result of the grouping. There are three queries that incorporate the use of a case statement in the select clause. Also, TPC-H queries tend to make use of subqueries that may represent a correlated (c) nested query, an uncorrelated (u) nested query, an inline view (i) or a traditional view (v).

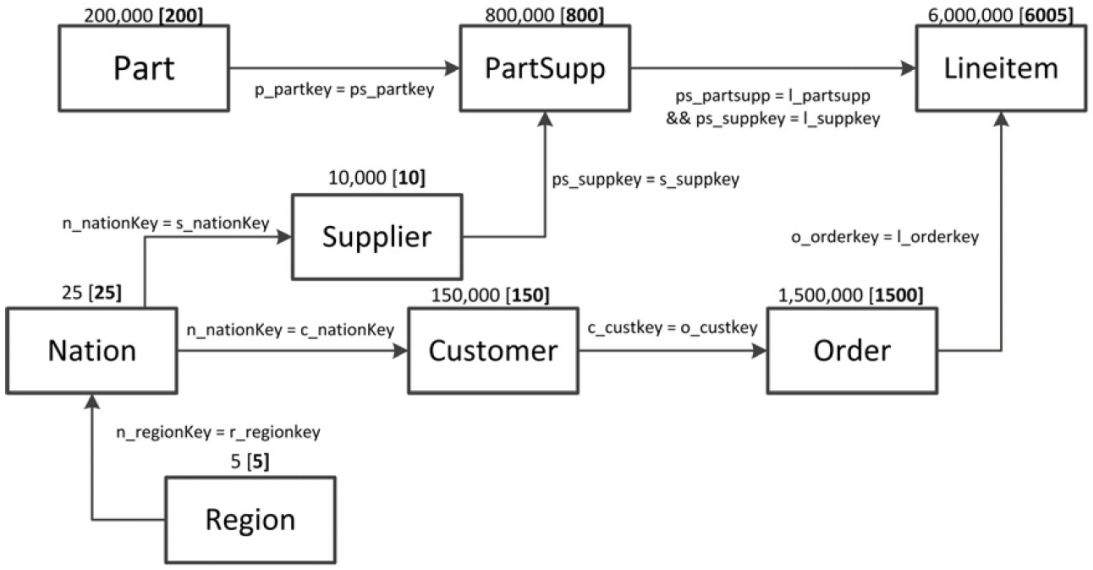


Figure 1. TPC-H Schema

	Join		Aggregation								Subquery			
Query	Inner	Outer	Avg	Count	Min	Max	Sum	Group By	Having	Case	c	u	i	v
Q1			✓	✓			✓	composite						
Q2	✓											✓		
Q3	✓						✓	composite						
Q4	✓			✓				simple			✓			
Q5	✓						✓	simple						
Q6							✓							
Q7	✓						✓	composite				✓	✓	
Q8	✓				✓		✓	simple		✓		✓	✓	
Q9	✓						✓	composite				✓	✓	
Q10	✓						✓	composite						
Q11	✓						✓	simple	✓			✓		
Q12	✓				✓		✓	simple		✓				
Q13		✓		✓				simple				✓	✓	
Q14	✓				✓		✓			✓				
Q15						✓	✓	simple				✓		✓
Q16	✓			✓				composite				✓		
Q17	✓		✓				✓				✓			
Q18	✓						✓	composite	✓			✓		
Q19	✓						✓							
Q20	✓						✓				✓	✓		
Q21	✓			✓				simple			✓			
Q22	✓		✓	✓			✓	simple			✓	✓	✓	

Table 1. Characterization of TPC-H Queries

LEARNING FROM BENCHMARK QUERIES

Being able to formulate SQL queries is a critical skill that may take students some time to master [13]. In fact, some of the most challenging topics to teach in a database course are grouping, aggregation, and nested queries [7]. From the 22 TPC-H queries, there are 16 queries that use grouping typically with aggregation. Through a code review of the benchmark queries, students can reinforce their knowledge of aggregate functions and grouping. In addition, TPC-H illustrates several advanced query constructs. Figure 2 shows two benchmark queries that illustrate inline views, outer joins, and case statements.

An inline view, which is sometimes called a derived table, is a subquery specified in the from clause. Unlike traditional views, which require create and drop statements, an inline view is temporary and visible only within the scoping rules of the query in which it is defined. Inline views are usually preferred over traditional views when the view is needed for only one query. Inline views also tend to be more efficient than creating temporary tables [11]. In addition, inline views can be used to include levels of aggregate functions into one query, which is illustrated in Query 13 in Figure 2. The business case of Query 13 is to provide a distribution of the number of orders that customers have placed, which are filtered to remove special categories. Query 13 uses an inline view to find each customer along with their number of orders. Using the computed order count per customer from the inline view, the outer query then performs an aggregation to specify the distribution of customers having an order count.

Query 13 also uses an outer join within the specification of the inline view to handle the case of customers with no orders. The outer join operator adds nulls while joining tables if there is no match [7]. Outer joins are useful when attempting to find missing values or when needed to display partially matched information [15]. In Query 13, the count of the null o_orderkey returns a result of 0 for those customers that do not have any orders. Note that the specification of the benchmark requires that one third of the customers do not place an order.

TPC-H Query 13

```
select c_count, count(*) as custdist
from  ( select c_custkey, count(o_orderkey)
        from customer left outer join orders
             on c_custkey = o_custkey
             and o_comment not like '%special%requests%'
        group by c_custkey) as c_orders (c_custkey, c_count)
group by c_count
order by custdist desc, c_count desc;
```

TPC-H Query 14

```
select 100.00 * sum(case when p_type like 'PROMO%'
                         then l_extendedprice * (1 - l_discount)
                         else 0
                         end) / sum(l_extendedprice *
                             (1 - l_discount)) as promo_revenue
from lineitem, part
where    l_partkey = p_partkey
         and l_shipdate >= '1995-09-01'
         and l_shipdate <  '1995-10-01';
```

Figure 2. TPC-H Queries 13 and 14 [14]

TPC-H also illustrates case statements, which are conditional statements that may be used in the select, group by or order by clause of a query. This operator can help provide a more informative type of result to the output of the query, group the query through specified ranges, or dynamically sort query results [8]. Query 14 uses a case statement to determine the percentage of revenue from promotional parts through a given date. The case statement only sums the revenue if the part type is promotional.

The benchmark queries provide an opportunity for students to practice query reading skills and to learn additional query constructs beyond what may be typically covered in an introductory database course.

TPC-H SUITE OF EXERCISES

A code review of the queries also provides a strong foundation for understanding the benchmark enterprise. Thus, the TPC-H database can be used as the basis of a homework assignment. Table 2 provides a list of eight queries that can be assigned to students for practicing their query writing skills over a realistic enterprise. These queries are nontrivial, but simpler than those provided in the benchmark. The goal is to represent useful business cases while covering various capabilities of SQL.

These sample queries can also be used to discuss the performance merits of alternative approaches. For example, Figure 3 provides four alternatives to the negation query, which finds the customers who have not placed any orders. This query can be expressed by using not in, not exists, except, or even an outer join. The class can discuss how the different versions of the queries can be executed by the database as written. However, the underlying query optimizer of a database product uses additional information, such as the size of the data, available indexes, and other data constraints, to create its query execution plan. For example, the various alternatives to executing a similar negation query in Oracle are discussed in [3], which also indicates a change in query execution plans across versions of the same product. Ultimately, the performance of a query in the application using a particular version of a database product must be tested and evaluated based on the requirements of the application.

Capability	Description	Query Schema
SPJ	Which parts are supplied by the EUROPE region? Display in ascending order by retail price.	(p_partkey, p_name, p_retailprice)
Min	For each part, find the minimum cost supplier. Display in ascending order by part number.	(p_partkey, s_suppkey, min_supplycost)
Max	Which customers have the maximum number of orders that contain any returned item? Display in ascending order by customer number.	(c_custkey, c_name, count_order)
Only One (Counting)	Which orders with an URGENT priority have only one lineitem? Display in ascending order by order number.	(o_orderkey, o_custkey, o_orderstatus)
Average	What is the average cost of purchases made by each customer in the AMERICA region? Display in ascending order by average order price.	(c_custkey, c_name, avg_orderprice)
Grouping (Sum/ Average)	Find all the parts in which the sum of its available quantity is less than the average available quantity of all the parts. Display in ascending order by part number.	(p_partkey, sum_availqty)
Negation	Which customers have no orders? Display in ascending order by customer number.	(c_custkey, c_name)
Division	For customers with an URGENT order, are all the parts on the order supplied by suppliers that are located in the same nation as the customer? Display in ascending order by customer number.	(c_custkey, c_nationkey, o_orderkey, o_orderstatus)

Table 2. Suite of Query Specifications

not in	outer join
select c.c_custkey, c.c_name from customer c where c.c_custkey not in (select o.o_custkey from orders o) order by c_custkey	select c_custkey, c_name from customer left outer join orders on customer.c_custkey = orders.o_custkey where orders.o_custkey is null order by c_custkey
not exists	**except (returns only c_custkey)**
select c_custkey, c_name from customer c where not exists (select o.o_custkey from orders o where o.o_custkey = c.c_custkey) order by c_custkey	select c.c_custkey from customer c except select o.o_custkey from orders o order by c_custkey

Figure 3. Negation query using different SQL specifications

The incorporation of the TPC-H benchmark enterprise with the suggested homework exercises was successfully incorporated in the most recent offering of a database course. Students indicated that they appreciated the "opportunity to work with and use a more interesting database." There were some additional advantages that were discovered. The benchmark gave students a nontrivial data definition example and illustrated the importance of documentation, since the students needed to explore the TPC-H document to understand the enterprise in its entirety and to look up constants used in the database in order to answer the queries.

DISCUSSION

Exposure to scalable performance benchmarks reinforces the SQL learned in an introductory database course. Through a case scenario, TPC-H not only stresses the need for performance, but also provides a way to expose students to other SQL techniques seen in the industry. Although there is a constant debate about the topics that should be incorporated into the curriculum, using an application for examples is a way to integrate new themes within existing courses [17]. Incorporating the TPC-H enterprise and benchmark queries into a database course will not only build upon the fundamentals students are required to learn at this level, but will also expose students to complex situations and operators often utilized in the industry.

ACKNOWLEDGEMENTS

This material is based upon work supported by the NSF under Grant No. CSR-0915325 and the New College Undergraduate Inquiry and Research Experience (NCUIRE) program at ASU.

REFERENCES

[1] Adams, E.S., Granger, M., Goelman, D., and Ricardo C., Managing the introductory database course: what goes in and what comes out? *SIGCSE Bull*, 36, 1(Mar. 2004), 497-498, 2004.

[2] Anderson, R., Jafar, M., and Abdullat, A., Teaching scalability issues in large scale database application development, *Information Systems Education Journal*, 5, 28 (Sept. 2007), 1-14, 2007.

[3] Burleson Consulting, Rewrite SQL Subqueries as Outer Joins, 2004, http://www.dba-oracle.com/oracle_tips_subq_rewrite.htm, retrieved January 3, 2012.

[4] Dietrich, S. W., Personal Web Page, http://www.public.asu.edu/~dietrich/, retrieved January 3, 2012.

[5] Dietrich, S. W., Urban, S. D., and Haag, S., Developing advanced courses for undergraduates: A case study in databases, *IEEE Transactions on Education*, Vol. 51, Issue 1, 138-144, 2008.

[6] Menasce, D. A., Software, performance, or engineering? *Proc. of the 3rd international workshop on Software and performance*, 239-242, 2002.

[7] Matos, V., Grasser, R., and Jalics, P., The case of the missing tuple: teaching the SQL outer join operator to undergraduate information systems students, *J. Comput. Sci. Coll.*, 22, 1 (Oct. 2006), 23-32, 2006.

[8] Mitchell, S., The power of SQL CASE statements, October 27, 2004, http://www.4guysfromrolla.com/webtech/102704-1.shtml, retrieved January 3, 2012.

[9] Murray M. and Guimaraes M., Expanding the database curriculum, *J. Comput. Sci. Coll.*, 23, 3 (Jan. 2008), 69-75, 2008.

[10] Nelson, D. A., Stirk, S., and Green, C., An evaluation of a diverse database teaching curriculum and the impact of research, *LTSN-ICS Teaching, Learning and Assessment in Databases Workshop*, 69-73, 2003.

[11] Papa, J., Data Points: Five Ways to Rev up Your SQL Performance, 2002, http://msdn.microsoft.com/en-us/magazine/cc301622.aspx, retrieved January 3, 2012.

[12] Pons, A. P., Database tuning and its role in information technology education, *Journal of Information Systems Education*, 14(4), 374-380, 2003.

[13] Prior, J. C., Online assessment of SQL query formulation skills, *Proc. of the fifth Australasian conference on computing education*, 20, 247-256, 2003.

[14] Transaction Processing Performance Council, TPC Benchmark H, Revision 2.14.3, 2011, http://www.tpc.org/tpch/, retrieved January 3, 2012.

[15] Viescas, J. L., and Hernandez, M. J., SQL Outer Join sample uses, 2007, http://searchsqlserver.techtarget.com/feature/SQL-OUTER-JOIN-sample-uses, retrieved January 3, 2012.

[16] Wagner, P. J., Shoop, E., and Carlis J. V., Using scientific data to teach a database systems course, *ACM SIGCSE*, 224-228, 2003.

[17] Walker, H. M., Curricular syncopations: When is a computing curriculum bloated? *ACM Inroads* 2, 2 (May 2011), 18-20, 2011.

[18] Wang, M., Guimaraes, M., and Myers, M. E., The database course (panel session): what must be taught, *ACM SIGCSE*, 403-404, 2000.

SENIOR PROJECT: GAME DEVELOPMENT USING GREENFOOT*

Karen Villaverde, Bretton Murphy
Computer Science Department
New Mexico State University
1290 Frenger Mall SH 123
Las Cruces, NM 88003
575-646-1609
kvillave@cs.nmsu.edu, hiei@nmsu.edu

ABSTRACT

In this paper we describe our very positive experience in teaching a senior project course-in one semester-with game development as the subject matter. In this course a 2D game engine called Greenfoot was used as the development platform. We describe why Greenfoot was chosen as our 2D game engine, what material was covered, and how the course was conducted. We also describe how the grading was performed, the quality of the students' game projects, features of Greenfoot that students liked for game development, features that students wished Greenfoot had, and future work.

INTRODUCTION

We always wanted to teach a senior project course where 2D game development was the focus. However, in our university, senior project courses are given in just one semester. Therefore, our main concern was whether the students would have enough time to learn to use a 2D game engine, and then design, prototype, develop, and test a complete senior project game in one semester. Giving the students an already familiar programming language was another problem. Because of these concerns we decided upon Greenfoot [6] as our game development platform.

In the spring of 2010, we taught a senior project course with Game Development as the subject matter and used Greenfoot as our development platform. The course was fourteen weeks long and was taught in the Computer Science department of New Mexico

---

State University. This was the first time that a senior project on game development has been taught in our department. We were very satisfied with the quality of the senior project games produced by the students. The students enjoyed the course very much, learned a lot, and felt a strong sense of accomplishment in having developed a complete 2D game-even though none of them had developed any game before.

In this paper, we describe our very positive experience in teaching this course with Greenfoot as our game engine. Greenfoot has been used to teach how to make very simple games to high school and university students [9], but to our knowledge nobody has used Greenfoot in a senior project course before. We hope that this paper encourages other faculty to give Greenfoot an opportunity as a powerful 2D game engine for senior project courses in the future.

GREENFOOT AS A GAME ENGINE FOR GAME DEVELOPMENT

Greenfoot is a free, 2D educational game engine developed in 2006 with the goal of teaching programming at high school level or above. It can be effectively used at school, college, and university levels, and even in advanced university courses. Greenfoot is a project of the University of Kent at Canterbury (UK) and Deakin University, Melbourne (Australia) with support from Sun Microsystems, Oracle, and Google. Greenfoot is a multiplatform game engine which runs on Windows, Mac OS, Linux, and Unix. Students write their games in Java. Greenfoot supports the full Java language and games can be run as applications, on a web browser, or on the Greenfoot environment.

Some of the best benefits of using Greenfoot is that Greenfoot has excellent resources for learning the engine (e.g., video tutorials by one of the creators of Greenfoot-a professor), Greenfoot is regularly updated, and the Greenfoot community is extremely active. The Greenfoot community posts, comments, ranks games in the Greenfoot gallery, participates in discussions, and asks/replies to questions in the Greenfoot forum. The Greenfoot gallery is where users can post their games and there are literally hundreds of them. They can be run directly on the website and the games' source code is available for download.

There are other game engines that use a general purpose language for 2D game development and have been used in educational settings, e.g., Microsoft XNA [11], DarkBasic [3], and Pygame [13]. However, most of these platforms have a very steep learning curve. The learning curve becomes even steeper if students have not programmed extensively in the platform's language. This makes them very challenging to learn for students in one semester while at the same time completing a senior project.

On the other hand, Greenfoot has an extremely shallow and straightforward learning curve with no major learning bumps. Students program their games in Java, a language that almost all computer science students know very well and have used it extensively by their senior year. This enables students to learn Greenfoot in a few weeks and gives them enough time to design, prototype, develop, and test a complete 2D game in the same semester.

REASONS FOR CHOOSING GREENFOOT

We chose Greenfoot as our 2D game engine for the following five reasons. First, our positive previous experience of using Greenfoot in another course. We taught a summer game programming course using Greenfoot as our 2D game engine the previous year. The only pre-requisite for this course was a data structures course in Java. Even though our summer semester is only 10 weeks, our students were able to learn the Greenfoot platform, learn how to use image and sound editors (Gimp [4] and Audacity [1]), and design, develop, and test two small but interesting 2D games. Therefore, we knew that is was possible for students to learn the Greenfoot platform in a short amount of time and develop some decent games in the same semester. Second, we wanted to try an experiment with our senior project students who love to play games but who have never actually developed a game before. Given our success with the games developed by our summer game programming students, we wanted to see the quality of games that could be produced by our senior project students working on teams. Third, we wanted a 2D game engine that could be learned in a short amount of time-a few weeks-so the students could have enough time to design, prototype, develop, and test one complete 2D game during our fourteen weeks course. Fourth, we wanted a 2D game engine with a general purpose language in which the students had programmed extensively (e.g., Java). Fifth, we wanted a free 2D game development platform where lots of different types of games could be implemented.

MATERIAL COVERED AND HOW THE COURSE WAS CONDUCTED

The main material that we covered in class came from the excellent videos tutorials and the Greenfoot manual available on the Greenfoot website. The video tutorials explain the use of the Greenfoot engine in a step by step and very educational way as they are written by the professor who created Greenfoot. This material was covered in a laptop equipped classroom with three projectors and lectures that were 100% hands on and extremely interactive.

Other material that we covered in class were game techniques such as color masking for collision detection, background scrolling, and handling levels to efficiently manage the different stages of a game; how to increase Greenfoot's memory heap in order to run large games; how to use Audacity and Gimp-including transparencies and scaling; and some game optimization techniques. We learned some of this material by studying some of the most popular games and demos in the Greenfoot website [2, 12]. We have now actually developed video tutorials that teach all these techniques and concepts step by step and are available at [8]. We also covered some material from the Greenfoot book [10]. Some of the material covered includes the use Greenfoot's helper classes SmoothMover and Vector for creating movement, how to add gravitational forces, apply gravity, how to handle proton waves for fire power, and differences between sound and image file formats and sizes.

The class met three days per week for a total of 185 minutes per week. In the first four weeks, we covered the use of the Greenfoot engine, the game techniques, the use of Audacity and Gimp, and the material from the Greenfoot book. Also, in the first four weeks while the students were learning the Greenfoot engine, they were given as homework to explore the Greenfoot gallery. They also brainstormed ideas for the type

of game, theme, and game play mechanics that they would like to develop for their senior project game. The students were free to select their own game type, game mechanics, theme, characters, sound effects, background music, etc. They were also free to make their own teams. We just made sure that each team had a good programmer. We had three teams of three students each.

The remaining ten weeks of the semester were spent on the design, prototype, development, and testing of the senior project games. The students did about half of their senior project work in our laptop equipped classroom while we supervised them, encouraged them, assisted them with their questions, and offered them suggestions. The decision for letting students work on their senior project in class worked extremely well to prevent student procrastination and greatly increased team communication. Also, the students appreciated the fact that we let them work in class because for them it was very difficult to meet outside class-all of our students had part-time and full-time jobs and some of them families of their own. However, our students did work (individually) on their project outside of class. During class, students divided among themselves some of the work so they could advance individually outside of class as well. The students were not required to complete software development documents other than the Java Doc documentation of all their code. This decision allowed our students to have extra time to spend for development, testing, and debugging of their games. Due to the time constraints of completing a game in one semester, the students followed a rapid application development approach. Our students developed their games in approximately 1850 minutes in class and 1800 minutes outside of class.

GRADING

Two criteria were used to grade the students: class attendance and participation (30%) and their senior game project (70%). The main criteria used in the evaluation of the senior game projects were quality, effort, creativity, Java Doc documentation of all their source code, as well as meeting the project deadlines: prototype version, alpha version, beta version, and final version. These project deadlines were evenly distributed in the last ten weeks of the semester. We observed that students worked extra hard when a deadline was approaching because they wanted to show that their game was the best. All the students were males and enjoyed proving that their game was the best during the games' presentations after each deadline.

GAME PROJECTS' QUALITY

All three senior projects were side-scrolling platformer games. Figures 1 through 3 show the three games developed by our students: Toby, Star Wars, and Retro All Stars.

In Toby [17] (Figure 1) the player plays the role of a robot called Toby. As Toby goes through the world, he must fight evil bees, turtles, birds, and snails that constantly attack him. At the same time Toby must avoid precipices and other obstacles. All of the art work was designed and created just for this game and consists of all unique work done by one of the team members who is an artist. One of the most interesting aspects of this

game is the parallax technique used in the scrolling background which is composed of several layered images. This technique allows images in the background to scroll slower the farther away they are from Toby, creating a kind of 3D effect. All the game elements-mechanics, programming, story, and esthetics-work together to form a single unified theme. The final product is a unique and very challenging game consisting of 36 different Java classes, 5582 lines of code, 90 sprites, and 16 sound effects. Toby is a great example of what Greenfoot can accomplish. The executable jar file is 17.8 MB.

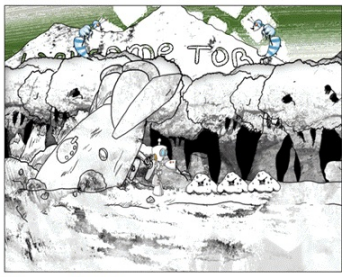


Figure 1. Toby

Figure 2. Star Wars

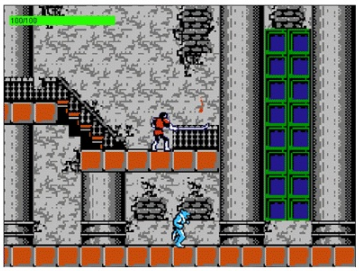


Figure 3. Retro All Stars

In Star Wars [16] (Figure 2) the player plays the role of Luke Skywalker who has to fight his way through the game destroying evil droids and stormtroopers. The player can use three different types of weapons: a light saber for close quarters combat, and a blaster pistol and machine gun for long range combat. There is also a force attack that can be used from a distance. The game consists of three levels. To complete a level, all enemies have to be defeated. The excellent character animation of this game enhances very well its game play and story. Star Wars consists of 28 Java classes, 4293 lines of code, 278 sprites, and 12 sound effects. The executable jar file without background music is 8.6 MB and with background music is 26 MB.

Retro All Stars [14] (Figure 3) is a parody game where the player plays the role of Castlevania's Simon. Game play is almost exact to that of the classic game Castlevania. But as Simon progresses through the game he finds himself in the worlds of Super Mario Brothers and Zelda fighting bosses from Mega Man, Teenage Mutant Ninja Turtles, and South Park's Chris Hansen (of NBC's Dateline). There are also references to Metal Gear Solid and lots of one-liners from Duke Nukem (including his rendition of the famous line from John Carpenter's "They Live"). Simon has to fight his way using only a whip as a weapon. Each level is progressed by making it to the end of the level in which case the next level begins immediately. The game consists of three levels. Simon dies when all his health is depleted. Retro All Stars consists of 30 Java classes and 3487 lines of code, 86 sprites, and 14 sound effects. The executable Jar file is 5.3 MB.

According to an in-class vote, the best game was Toby, followed by Star Wars, and then Retro All Stars. The three games where voted for in the Greenfoot gallery by the Greenfoot community and obtained the same rankings as the in-class vote. All three games were among the best 10 games in the Greenfoot gallery at one time and received lots of comments by the Greenfoot community. Toby was chosen as a Greenfoot showcase scenario by the Greenfoot creators.

FEATURES OF GREENFOOT THAT STUDENTS LIKED

Table 1 below lists all the features of Greenfoot that our students liked.

Greenfoot features that students liked
(1) Greenfoot has a very clean and simple user interface that is easy, fast, and intuitive to learn.
(2) Greenfoot's 2D world grid can be adjusted to any number of cells with each cell adjusted to any number of pixels down to a single pixel size.
(3) Greenfoot's user interface includes a class diagram browser which provides a very nice way to view all the game's classes and their inheritance relations.
(4) Greenfoot supports Java Doc and it is very easy to switch a file's format-in the Greenfoot's code editor-between Java code and Java Doc by just clicking a button.
(5) Greenfoot has an extremely easy to use and very complete Integrated Development Environment (IDE) where class diagram browsing, coding, compilation, debugging, execution control, and accessing Java Doc documentation of user's code-as well as the Greenfoot API and Java library documentation-are very pleasant to use.
(6) Greenfoot's API is small-it consists of only six classes, with a relatively modest number of methods, easily learnable, and very well documented in Java Doc format.
(7) Greenfoot's API provides very useful collision detection methods, e.g., there is a collision method that returns a list of all objects that intersect a given object.
(8) Greenfoot's API provides very useful methods for handing keyboard and mouse input.
(9) There are several very useful support classes in the Greenfoot website for animations, explosions, smooth movements, handling vectors, and GIF animations.
(10) Greenfoot programs can be paused, single stepped, speed up, and slow down at run-time using a scroll bar.
(11) Objects can be manually added to a game by right clicking their class in the class diagram.
(12) An object's methods can be executed by right clicking an object and selecting one of its methods for execution.
(13) An object's instance variables and their values can be examined at run time.
(14) Greenfoot provides automatic thread management. Greenfoot executes the main method-act method in Greenfoot's terminology-of each object in a round robin fashion. And even though it is not possible to specify the order of execution of individual objects, it is possible to specify the order of execution for classes of objects.

(15) Greenfoot's performance is good in that games with several hundreds of objects, all checking collisions between each other, run smoothly.
(16) Greenfoot handles most computer graphics on its own, leaving the programmer to concentrate on the game logic.

Table 1. Features of Greenfoot that students liked

FEATURES STUDENTS WISHED GREENFOOT HAD

Even though the students enjoyed working with Greenfoot (version 1.5.6) very much, they did express their desire for Greenfoot to have the following features. Fortunately, most of these problems have been fixed in Greenfoot version 2.0.

Greenfoot features wished by students
(1) The Greenfoot code editor did not have code completion, scope highlighting, and navigation view features. These features have been implemented in version 2.0.
(2) Greenfoot does not have version control. The students used Google wave to keep track of their game versions.
(3) The Greenfoot tracing and debugging feature that our students used did not work properly so the feature was not used at all and had to improvise to trace and debug their games. This feature now works very well in Greenfoot version 2.0.
(4) Greenfoot did not have mp3 or midi support and wav support was not very good. We had to code wav and midi players for our students to use. Fortunately, Greenfoot 2.0 now has good mp3, wav, and midi support.
(5) The Greenfoot game engine had some annoying bugs suspected to be memory leaks. Most of these bugs have been removed in version 2.0.
(6) Greenfoot does not do game optimizations by itself. All optimizations must be planned, studied, and programmed.
(7) The Greenfoot gallery only accepts games of 20MB or less. Fortunately, there is a way around this restriction when posting big games: students can post the URL to their own websites where their game is located and can be played as well. Unfortunately, most Greenfoot users do not click on these URLs-they prefer to play games that can be played directly on the Greenfoot gallery.

Table 2. Greenfoot features wished by students

FUTURE WORK

In the future we plan to include the teaching of game design concepts in this course. We plan to use the book The Art of Game Design [15] which has been proven very useful in a game design and development course that we just finished teaching. We believe that the knowledge from this book will encourage, help, and guide students to design and develop innovative games that have better flow channels (skills vs. challenges

progressions), better interest curves (interest vs. time progressions), polished game play mechanics, well designed interfaces, and good game balancing.

We plan to require students to complete a design and optimizations document where they study, plan, and analyze the memory and time complexities of their data structures and algorithms. The students will also study and plan as many optimizations as possible. We felt that this document was very much needed to avoid game performances from lagging. We also plan to require students to learn and use a code versioning program like Github [5]. This is a very useful skill that they should know before they graduate.

In order to achieve all the above, we will require students to learn Greenfoot at home instead of us teaching it in class. It is indeed possible for students to do this due to the excellent video tutorials in the Greenfoot website and the additional video tutorials that we have created since we taught this class [8].

REFERENCES

[1] Audacity, http://audacity.sourceforge.net, retrieved December 22, 2011.

[2] Biomekanoid Chicken, http://greenfootgallery.org/scenarios/338, retrieved December 22, 2011.

[3] Dark Basic, http://www.thegamecreators.com, retrieved December 22, 2011.

[4] Gimp, http://www.gimp.org, retrieved December 22, 2011.

[5] Github, https://github.com, retrieved December 22, 2011.

[6] Greenfoot, http://www.greenfoot.org, retrieved December 22, 2011.

[7] Greenfoot Senior Project, http://www.cs.nmsu.edu/~kvillave/GreenfootSP, retrieved December 22, 2011.

[8] Greenfoot Video Tutorials, http://www.cs.nmsu.edu/~kvillave/GreenfootVideos, retrieved December 22, 2011.

[9] Kolling, M. The Greenfoot programming environment, *ACM Transactions on Computing Education*, 10, (4), Article 14, 2010.

[10] Kolling, M. *Introduction to Programming with Greenfoot*, Upper Saddle River, NJ: Prentice Hall, 2010.

[11] Microsoft XNA, http://create.msdn.com, retrieved December 22, 2011.

[12] Polle's Sidescroll Demo, http://greenfootgallery.org/scenarios/236, retrieved December 22, 2011.

[13] Pygame, http://pygame.org, retrieved December 22, 2011.

[14] Retro All Stars, http://greenfootgallery.org/scenarios/1397, retrieved December 22, 2011.

[15] Schell, J. *The Art of Game Design*, Burlington, MA: Morgan Kaufmann Publishers, 2008.

[16] Star Wars, http://greenfootgallery.org/scenarios/1399, retrieved December 22, 2011.

[17] Toby, http://greenfootgallery.org/scenarios/1405, retrieved December 22, 2011.

RELATING AUTOMATA TO OTHER FIELDS*

Pradip Peter Dey, Mohammad Amin, Gordon W. Romney, Bhaskar Raj Sinha,
Ronald F. Gonzales, Alireza Farahani and Hassan Badkoobehi
National University, School of Engineering, Technology, and Media
3678 Aero Court, San Diego, CA 92123, U.S.A.
{ pdey, mamin, gromney, bsinha, rgonzales, afarahan, hbadkoob}@nu.edu

ABSTRACT

Automata classes including Turing Machines, Pushdown Automata and Finite Automata define the most elegant models of computation in terms of set processors. This study elaborates pedagogically motivated intuitive and formal relations between mathematical models and other computer science areas, so that students can relate different areas of computer science in a meaningful way. Cases that promote learning about theoretical models are presented with other related areas, such as programming languages, compilers and software design. Dynamic aspects of software can be appropriately modeled by certain automata based models. Visualizations are developed to help students in their initial stages of understanding of these relations. The visualizations demonstrate that mathematical models such as Pushdown Automata are reasonable processors of some programming language features such as balanced {'s and }'s which are evidently helpful in learning this kind of relation.

INTRODUCTION

Turing Machines (TMs), two-stack Pushdown Automata (2PDA), Linear Bounded Automata (LBA), Pushdown Automata (PDA), Finite Automata (FA) and Non-deterministic FA (NFA) are some of the most elegant mathematical models that are taught in automata theory classes. These models of computation define computability in clear terms and provide scope and limitations of computer science in revealing ways. There are some excellent textbooks on automata theory or theory of computation [4, 6, 9, 10, 11, 12, 14, 17, 19]. Following these books one can teach automata theory as a self-contained course. Some of the books, such as [9, 14], give considerable attention to

---

applications of automata. However, an instructor needs to decide about a number of related questions: (a) To what extent these automata should be related to other areas of computer science in teaching computer science classes? (b) Is there any advantage in relating automata classes to other areas? (c) What are the best strategies to relate automata theory to other areas of computer science? We struggle with questions such as these and try to find reasonable answers. Almost every book on automata theory refers to some other fields such as compilers, interpreters, programming languages, data structures and software engineering. However, most of the time automata theory students remain busy with proving theorems of equivalent classes, pumping lemmas and so on. When a course is taught, the focus of the course goes to the central ideas. Thus, when a course on programming languages is taught, usually not enough attention is given to PDA. This paper investigates the ways in which courses on different areas of computer science can be taught by relating certain aspects to benefits students. The remainder of this paper discusses a viable alternative way of teaching some related courses in computer science. It also suggests that instead of teaching a traditional course on automata theory, a new course on "Automata and Related Topics" should be taught with tools and strategies that relate the automata to other subjects.

TOOLS AND STRATEGIES FOR RELATING AUTOMATA TO OTHER FIELDS

Although applications of automata are described in various details in most textbooks [4, 6, 9, 10, 11, 12, 14, 17, 19], additional steps need to be taken to relate automata to other subjects. Some static and dynamic visualization tools can be considered for expanding the applications as relations among the subjects. A special case of the relation between PDA and programming language processing can be considered, where a balanced number of {'s and }'s needs to be processed. This case is presented with a non-regular Context-Free Language, L_m = { $\{^n c\}^n$: where n >= 0 }. It is to be noted that L_m has strings with matching number of {'s and }'s separated by a **c** . That is, strings of L_m are of the form: **c, {c}, {{c}}, {{{c}}}, {{{{c}}}},** . . . . What is interesting about L_m is that it has a string pattern that is similar to that of programming languages such as Java and C++. In fact, syntactic structures of a programming language are defined by Context-Free Grammars (CFGs) in a way that is similar to that of the CFG of L_m given below:

S ➔ {S} | c

This CFG generates or derives a balanced number of {'s and }'s. PDA are designed to accept languages with strings that have similar patterns. That is, a Pushdown Automaton will accept strings like **{c}, {{c}}, {{{c}}}** . . . . Pushdown Automata use a stack data structure for matching equal number of {'s and }'s without counting them. A stack is an interesting data-structure which allows operations such as push and pop and increases or decreases its stored contents in a Last-In-First-Out (LIFO) manner. Stacks are used in PDA for processing CFL's as described in textbooks [4, 6, 9, 10, 11, 12, 14, 17, 19]. One needs to consider multiple ways of presenting automata to students in order to highlight their formal and intuitive relations to other fields. PDA can be presented in various ways including state diagrams. In Figure 1, a PDA for L_m = { $\{^n c\}^n$: where n >= 0 } is presented visually as a finite set of states connected with transitions based on the notations given in [9] with minor adjustments that show the

stack explicitly with the bottom of the stack on the left, and define transitions with the pair: R,T/TP where R is the symbol read from the input, T preceding / is the topmost stack symbol before the transition is taken, TP following / is the sequence of topmost stack symbol(s) after the transition is taken and P is an optional symbol which appears only with "push transitions". The state diagrams for PDA given in [7] do not explicitly show the stack.

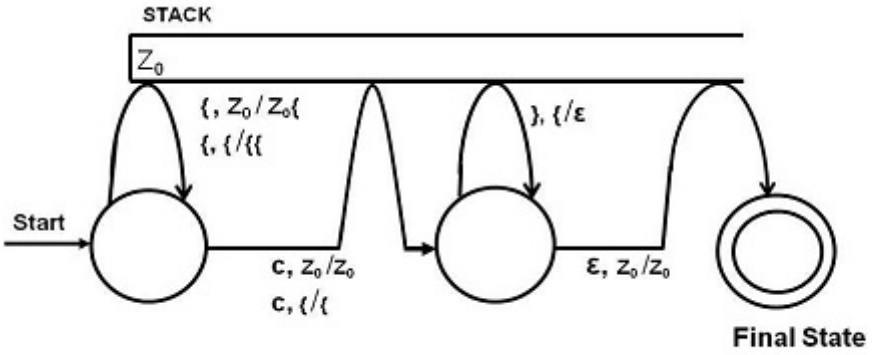


Figure 1: A Pushdown Automaton for L_m = { $\{^n c\}^n$: where n >= 0 }.

Ordinarily, static visualizations of PDA can be done with a sequence of state diagrams, such as the one given in Figure 1. One type of dynamic visualization is shown in the form of an animation on the following web site: www.asethome.org/pda. Our approach is based on the pioneering work of Rodger in the area of visualization of automata [15-16]. Some recent studies have criticized dynamic visualizations compared to that of static ones [13, 20], which does not apply to our visualizations, because these are not comparable to static ones.

Suppose a string like **{{c}}** is given as an input to the PDA of Figure 1. Then, the machine starts at the start state and scans the first { from the input and pushes a { into the stack by taking the transition marked by, **{ , Z_0 / Z_0{** . The meaning of this transition label is "when reading a { and the stack is empty (marked by Z_0) push a **{** onto the empty stack (marked by Z_0)". Then it consumes the next **{** from the input by taking the same loop with the transition marked by **{, { /{{** . Next, it consumes the symbol c by taking the transition marked by **c, { /{** which means "read a c from the input when there is a **{** on top of the stack and leave the stack unchanged". Next, it reads the fourth symbol, **}** , from the input and pops a **}** from the stack taking the transition marked by **}, { /□** . Then, it scans the next **}** by taking the same transition marked by **}, { /□** again. Then, it reaches the final state by taking the transition marked **□, Z_0 /Z_0** . At that moment the stack is empty and the entire input is consumed and therefore the input **{{c}}** is accepted by the machine. The PDA given above accepts any string with a sequence of {'s followed by a c followed a number of **}**'s that balances **{**'s. That is, strings such as **c, {c}, {{c}}, {{{c}}},** . . . . are accepted by the machine. An input is accepted by a PDA if all of the following conditions are met simultaneously: (a) the input is entirely consumed, that is, no other symbols left in the input; (b) the machine is in a final state; (c) the stack is empty. In a survey, 19 out of 26 respondents (73%) found the dynamic visualization of PDA helpful in learning the relation between PDA and programming language features.

In addition to the connection between PDA and syntactic analysis of programming languages, the relations among lexical analyses, FA and NFA need to be demonstrated. An animation of an NFA for a programming language lexical analysis [9, 18] is given on a page linked to www.asethome.org/automata. It rejects identifiers that start with a digit and accepts other well-defined identifiers. In addition, programming assignments can be developed based on FA or NFA, as shown in some examples linked to the above website. Model checking is another important area of application of automata [3].

Software development relies on modeling the software in various levels, including the design level. Design tools based on statecharts [7-8] have been very useful for modeling dynamic aspects of software. Statecharts are basically TMs presented in a notation that is appropriate for representing software features in an intuitive way. A statechart diagram representing the dynamic aspects of computing average family size of a town is given in Figure 2; an experimental implementation of the software in an applet can be found at http://www.asethome.org/automata/soft/average.html.

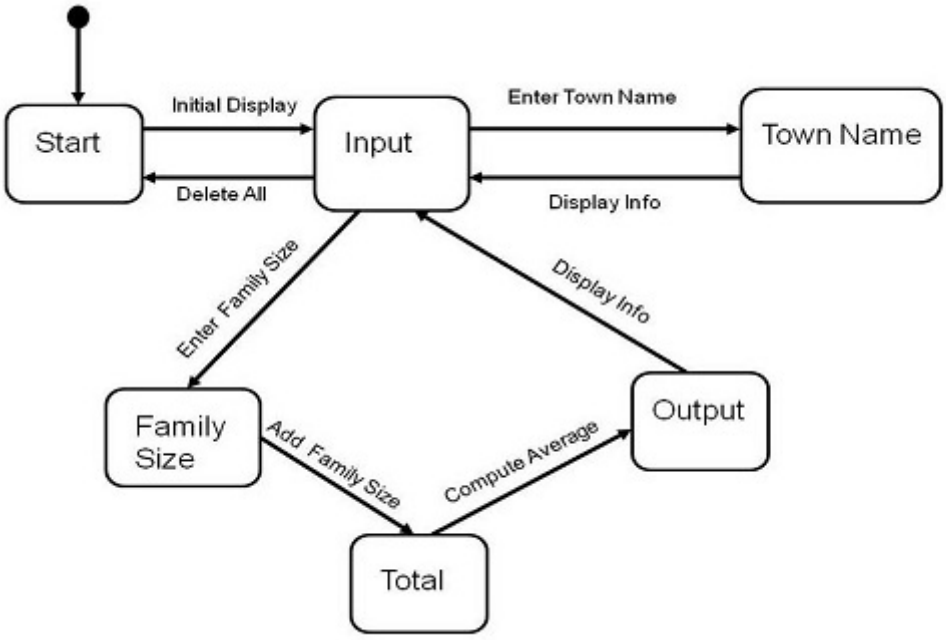


Figure 2: A Statechart Diagram for computing average family size

The relationships among automata, software engineering, compilers, model checking, data structures and programming languages can be best understood in a course on "Automata and Related Topics" at the undergraduate level. A description for this course is not given here, because the description should be based on the needs of the program in a college. Either [9] or [14] can be used as the main text for the course along with supplemental materials such as [15]. Examples of applications described in [9] and [14] are very helpful for teaching purposes. The course may use agile problem driven teaching [5] or problem-based learning [1-2] or game-based learning [21] for effective teaching [22]. Agile Problem Driven Teaching is closely related to problem-based learning [1-2]; however, it combines problem-based free inquiry with direct instructions in an agile process in order to achieve the course learning outcomes. Visualization

clarifies concepts and supports problem solving. The dynamic visualization of PDA, mentioned earlier, was introduced to the students first in the context of problem solving, although it was always available at the web site.

CONCLUDING REMARKS

There is no doubt that several courses, including compilers, programming languages, data structures and software engineering can be related to automata theory. These courses should be taught using the tools and strategies that relate them well. A new course on "Automata and Related Topics" can be taught using agile problem driven teaching strategies along with static and dynamic visualizations. There are advantages in relating topics of different courses together in order to teach them with reasonable interpretations and applications. Future research may concentrate on collecting and analyzing data for measuring teaching effectiveness and students' satisfaction in this area.

REFERENCES

[1] Barell, J., *Problem-based learning: An Inquiry Approach*, Corwin Press, 2nd Edition, 2006.

[2] Barrows, H. S., *How to design a problem-based curriculum for the preclinical years*. New York: Springer, 1985.

[3] Clarke, E., Grumberg, O., Peled, D., *Model Checking*, MIT Press, Cambridge, 1999.

[4] Cohen, D., *Introduction to Computer Theory*, (2nd ed.), John Wiley & Sons, 1997.

[5] Dey, P., Gatton, T., Amin, M., Wyne, M., Romney, G., Farahani, A., Datta, A., Badkoobehi, H., Belcher, R., Tigli, O., Cruz, A., Agile Problem Driven Teaching in Engineering, Science and Technology. In *the Proceedings of the American Society for Engineering Education-Pacific Southwest 2009 conference* ASEE-PSW 2009, San Diego, California, U.S.A., March 19-20, 2009. http://www.asethome.org/asee/ASEE_PSW_2009_Proceedings.pdf

[6] Goddard, W., *Introducing the Theory of Computation*, Jones & Bartlett, 2008.

[7] Harel, D., Statecharts: A visual formalism for complex systems, *Science of Computer Programming*, v.8 n.3, p.231-274, 1987

[8] Harel, D., Politi, M., *Modeling Reactive Systems with Statecharts: The Statemate Approach*, McGraw-Hill, 1998.

[9] Hopcroft, E., Motwani, R., Ullman, D., *Introduction to Automata Theory, Languages, and Computation*, Pearson Education, 2007.

[10] Kinber, E., Smith, C., *Theory of Computing: A Gentle Introduction*, Prentice Hall, 2001.

[11] Kozen, D., *Theory of Computation*, Springer, 2006.

[12] Lewis, H., Papadimitriou, C., *Elements of the Theory of Computation*, Prentice Hall, 1998.

[13] Lowe, R., Schnotz, W., *Learning with Animation: Research Implications for Design*, Cambridge University Press, 2007.

[14] Rich, E., *Automata, Computability and Complexity: Theory and Applications*, Prentice Hall, 2007.

[15] Rodger, S. H., Finley, T. W., *JFLAP: An Interactive Formal Languages and Automata Package,* Jones & Bartlett Publishers, 2006.

[16] Rodger, S.H., Bressler, B., Finley, T., Reading, S., Turning automata theory into a hands-on course. SIGCSE 2006: 379-383., 2006.

[17] Sakarovitch, J., *Elements of Automata Theory*, Cambridge University Press, 2009.

[18] Sebesta, R., *Concepts of Programming Languages*, 9th Ed., Addison-Wesley, 2009.

[19] Sipser, M., *Introduction to the Theory of Computation*, PWS Publishing, 2006.

[20] Tversky, B., Morrison, J., Betrancourt, M., Animation: Can It Facilitate? *International Journal of Human Computer Studies,* Volume 57, 247-262, 2002.

[21] Van, R. E., Game-based learning. *EDUCAUSE Review*, 41(2), 16-30, 2008.

[22] Wallis, C., How to make great teachers? *Time*, 171 (8), 2008.

DATABASE ANIMATIONS FOR MANY MAJORS*

CONFERENCE TUTORIAL

Suzanne W. Dietrich
Mathematical & Natural Sciences
Arizona State University
Phoenix, AZ 85069-7100
602-543-5628
dietrich@asu.edu

Don Goelman
Department of Computing Sciences
Villanova University
800 Lancaster Avenue
Villanova, PA 19085
610-519-7346
don.goelman@villanova.edu

This tutorial demonstrates and discusses two animations designed for teaching fundamental database concepts to many majors. The first animation provides an introduction to relational databases, and the second covers querying relational databases. The objective is for educators across many disciplines to incorporate the animations into their existing courses to fit their pedagogical needs. The development of these customizable FLASH animations is part of a collaborative grant funded by the National Science Foundation for reaching out to many majors using databases, entitled "Databases for Many Majors: A Student-Centered Approach".

PROVEN STRATEGIES THAT INCREASE PARTICIPATION OF HIGH SCHOOL STUDENTS IN COMPUTING*

CONFERENCE TUTORIAL

Renee L. Ciezki
Estrella Mountain Community College
3000 North Dysart Road
Avondale, Arizona 85392
623-262-5307
renee.ciezki@estrellamountain.edu

Much has been written about the decrease in the number of students pursuing computing in colleges. Since most students develop an opinion about computing before reaching college, it is beneficial for faculty who want to increase their enrollment to do outreach aimed at a younger audience. To help encourage high-school/university interactions, the Development Committee of the Advanced Placement Computer Science A Course has proposed and organized this and similar sessions for several regional and national meetings. This session will present ideas on how colleges can work with high school teachers and others to promote computer science to high schools students. This session will proceed in two main sections: 1) Ideas that have worked for members of the Advanced Placement Computer Science A Development Committee, and 2) Brainstorming with session attendees to explore alternative ideas for outreach and to identify further approaches.

INDEX OF AUTHORS

JCSC Volume 27 Number 4 April 2012

Muhlenberg College
2400 Chew Street
Allentown, PA 18104-5586